1992

A mixed excitation vocoder with fuzzy logic classifier

Moore, James Thomas

Monterey, California. Naval Postgraduate School

Approved for public release; distribution is unlimited.

A MIXED EXCITATION VOCODER WITH FUZZY LOGIC CLASSIFIER

by

James Thomas Moore
Lieutenant, United States Coast Guard
B.S., United States Coast Guard Academy, 1984

Submitted in partial fulfillment of the
requirements for the degree of

ELECTRICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL

June 1992

UNCLASSIFIED
CURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

REPORT SECURITY CLASSIFICATION NCLASSIFIED	1b RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
DECLASSIFICATION / DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
NAME OF PERFORMING ORGANIZATION aval Postgraduate School	6b. OFFICE SYMBOL (If applicable) EC
7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
ADDRESS (City, State, and ZIP Code) onterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
NAME OF FUNDING / SPONSORING ORGANIZATION	8b OFFICE SYMBOL (If applicable)
9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

TITLE (Include Security Classification)
MIXED EXCITATION VOCODER WITH FUZZY LOGIC CLASSIFIER

PERSONAL AUTHOR(S)
OORE, James Thomas

TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
gineer's Thesis	FROM _____ TO _____	1992 June	93

SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the thor and do not reflect the official policy or position of the Department of Defense or the US Government.

COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Fuzzy Logic; Fuzzy Systems; Speech Coding; Vocoders; Linear Predictive Coding; Signal Classification

ABSTRACT (Continue on reverse if necessary and identify by block number)

he aim of this thesis work is to explore the use of fuzzy systems in a peech coding and classification application. A mixed excitation LPC ased speech coder is developed. The excitation classifier for the speech oder is then implemented using a fuzzy system. The fuzzy logic based lassifier determines the type of excitation to be used in constructing he synthetic speech. The results of various implementations of this peech coder are presented for comparison. This work demonstrates that fuzzy system can be developed and implemented for a classification roblem.

DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
NAME OF RESPONSIBLE INDIVIDUAL TUMMALA, Murali	22b TELEPHONE (Include Area Code) 408-646-2645 / 22c OFFICE SYMBOL EC/Tu

Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603 — UNCLASSIFIED

ABSTRACT

The aim of this thesis work is to explore the use of fuzzy systems in a speech coding and classification application. A mixed excitation LPC based speech coder is developed. The excitation classifier for the speech coder is then implemented using a fuzzy system. The fuzzy logic based classifier determines the type of excitation to be used in constructing the synthetic speech. The results of various implementations of this speech coder are presented for comparison. This work demonstrates that a fuzzy system can be developed and implemented for a classification problem.

THESIS DISCLAIMER

The computer programs developed during the course of this research were used to illustrate the validity of the proposed ideas. The speech coders were tested using widely varying speakers and for different operating parameters, but they have not undergone an exhaustive testing procedure. A reasonable effort has been made to eliminate computational and logic errors, but the programs should not be considered fully verified. Any application of these programs without additional verification is at the user's risk.

Sun Workstation and SOUNDTOOL are registered trademarks of Sun Microsystems, Inc. MATLAB is a trademark of The MathWorks, Inc.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The aim of this thesis work is to explore the use of fuzzy systems in a speech coding and classification application. The speech coder selected was a mixed excitation LPC coder. The excitation consists of a mixture of high pass filtered noise and low pass filtered pulses. The ratio of pulse amplitude to noise variance is determined based on the voiced power of the speech. The placement of pulses in the excitation is based on the classification of the analysis frame. The classification task is carried out by a fuzzy logic based algorithm.

There are several advantages to using the fuzzy system for classification. Among them are:

- The classification thresholds can be made to adapt to different speakers and changes in speaking patterns.
- The classification rules can be formulated in linguistic terms, e.g., IF energy is HIGH and zero crossing rate is LOW, THEN the frame is VOICED. The structure of the fuzzy system then translates these rules into a numeric representation.
- Input parameters and classification rules can be easily changed to allow the testing of new configurations. Rules can also be adaptively generated using training data and a learning algorithm.

The thesis is organized as follows. Chapter II presents background material on fuzzy logic and fuzzy systems. Chapter III reviews various types of LPC based speech coders. Chapter IV shows the development of the deterministic model of the mixed

excitation speech coder. Performance results of several different implementations are presented. Chapter V presents the development of the fuzzy classifier and presents results from several implementations. Chapter VI lists conclusions and suggestions for further reasearch.

II. AN OVERVIEW OF FUZZY SYSTEMS

A. Introduction

Fuzzy logic developed as an outgrowth of research into the Heisenberg uncertainty principle and logical paradoxes in the 1920s and 1930s. In a mathematical sense, fuzziness means multivaluedness or multivalence. Simply put, fuzzy theory holds that all things are a matter of degree. The idea of multivalued sets was formalized into a comprehensive mathematical framework and given the name fuzzy sets by L. A. Zadeh in 1965 [Ref. 1].

Although the theoretical basis for fuzzy theory goes back to the early part of this century, it was only recently that fuzzy theory was applied to commercial areas [Ref. 2: p. 18]. Fuzzy systems use fuzzy logic to describe relationships between input and output variables. Much of the application development work has been done in Japan; however, fuzzy system applications are gaining wider acceptance in the West. The majority of these applications have been in the area of control theory.

The power of fuzzy systems is that they "reason" with parallel associative inference. This parallel architecture is easily implemented on a VSLI chip or by using optical devices. The most commonly implemented fuzzy system is the fuzzy associative memory (FAM).

B. Fuzzy Sets

It is helpful to compare fuzziness to randomness to gain a feel for the nature of fuzzy sets. Fuzziness describes event ambiguity. It measures the degree to which an event occurs, not whether it occurs. Randomness describes the uncertainty of

an event occurrence. Either the event occurs, or it does not. To illustrate, the probability of rain tomorrow describes a random event. The fuzziness of the event or the degree of rain can be described as light, moderate, or heavy. Fuzzy theory can be shown to contain probability theory as a limiting case [Ref. 2: p. 291].

1. Geometry of Fuzzy Sets

The key to reasoning with fuzzy sets is the concept of membership. A membership value $m_A(x)$ describes the degree to which element x belongs to set A. For the discrete set, $\mathbf{X} = \{x_1, \ldots, x_n\}$, membership in set A is described by a membership function. The domain of this function is $\mathbf{X} = \{x_1, \ldots, x_n\}$, and the range is $[0, 1]$ so that the function describes a mapping $m_A : \mathbf{X} \rightarrow [0, 1]$. The fuzzy power set of $\mathbf{X}$ is then defined as the set containing all possible subsets of $\mathbf{X}$ and is denoted as $F(2^X)$. Kosko uses a geometrical framework to illustrate the nature of fuzzy sets [Ref. 2: pp. 269-275]. The fuzzy power set of $\mathbf{X}$ is represented as a unit hypercube $I^n = [0, 1]^n$. A fuzzy set is then a point within the hypercube where the exact location of the point is described by a fit vector. The fit vector of A indicates the membership of each element of $\{x_1, \ldots, x_n\}$ in A. The vertices of the hypercube then represent nonfuzzy subsets, and the midpoint of the hypercube represents the maximally fuzzy (i.e., the most ambiguous) set. Within this context, fuzzy set operators and fundamental theorems are developed.

The basic fuzzy set operators are defined as:

$$\text{intersection}: \quad m_{A \cap B} = \min(m_A, m_B) \tag{2.1}$$

$$\text{union}: \quad m_{A \cup B} = \max(m_A, m_B) \tag{2.2}$$

$$\text{complementation}: \quad m_{A^c} = 1 - m_A. \tag{2.3}$$

Although these definitions appear similar to nonfuzzy set operations, it is important to note a major difference. In order for a set to be fuzzy, there must be some degree

of ambiguity. To represent this ambiguity, a basic tenet of nonfuzzy set theory must be violated, namely, A is properly fuzzy iff $A \cap A^C \neq \emptyset$ and $A \cup A^C \neq \mathbf{X}$.

The size of a fuzzy set is measured by a quantity known as the cardinality or sigma-count or simply the count, $M(A)$. The count of A is the sum of the fit values or

$$M(A) = \sum_{i=1}^{n} m_A(x_i). \tag{2.4}$$

This is an extension of the simplest distance measure between fuzzy sets, the ℓ_1 or fuzzy Hamming distance. This distance is defined as the sum of the absolute fit differences. This relationship is easily shown to be $M(A) = \ell_1(A, \emptyset)$.

2. Fuzzy Entropy Theorem

The fuzziness of a set is measured by a fuzzy entropy measure. In information theory, entropy describes the uncertainty of a system or message. In fuzzy set theory, a fuzzy set describes the system or message, and its uncertainty equals its fuzziness.

Within the geometrical framework of the unit hypercube representation, the fuzziness of a set is determined by the distance from it to the nearest vertex. A nonfuzzy set is located at a vertex, and a maximally fuzzy set is located at the midpoint of the hypercube. Therefore, a fuzzy set located at a vertex has zero entropy, and a fuzzy set located a the midpoint has maximum entropy. This idea can be expressed by defining the distance between the fuzzy set, A, and the nearest vertex as a and the distance between A and the farthest corner as b. The entropy is then the ratio a/b. The distance a is equivalent to $M(A \cap A^C)$ while b is equivalent to $M(A \cup A^C)$ [Ref. 2: p. 276]. Thus, fuzzy entropy is defined as:

$$E(A) = \frac{M(A \cap A^C)}{M(A \cup A^C)}. \tag{2.5}$$

With the information theory entropy measure, a sure event conveys minimum information and has zero entropy while an impossible event conveys maximum information and has infinite entropy. Equation 2.5 defines entropy in a similar manner as is seen by considering an event x described by fit value f. If the event is maximally fuzzy or maximally ambiguous, $f = 1/2$ and $E(f) = E(1/2) = 1$; thus, x would be maximally informative. Conversely, if the event is clear or unambiguous, then it is minimally informative and $f = 0$ or $f = 1$ and $E(f) = E(0) = E(1) = 0$.

3. Subsethood Theorem

Earlier, the fuzzy power set of the input domain $\mathbf{X}$ was defined as $F(2^X)$, the set containing all subsets of $\mathbf{X}$. Geometrically, this was visualized as the unit hypercube I^n. This idea of power sets extends to fuzzy sets as well. The power set of fuzzy set B, $F(2^B)$, is the set of all subsets of B. $F(2^B)$ then defines a hyperrectangle within the unit hypercube. It has one vertex on the origin, and its side lengths are equal to the fit values of B. With this in mind, the degree to which a fuzzy set A is a subset of fuzzy set B is defined as the subsethood of A to B, $S(A, B)$. Mathematically this is:

$$S(A, B) = \text{degree}(A \subset B) \tag{2.6}$$

$$= m_{F(2^B)}(A). \tag{2.7}$$

Subsethood is usually put in a more workable form as:

$$S(A, B) = \frac{M(A \cap B)}{M(A \cup B)} \tag{2.8}$$

with the following corollaries:

1. $$0 \leq S(A, B) \leq 1 \tag{2.9}$$

2. $$S(A, B) = 1 \text{ iff } A \subset B \tag{2.10}$$

3. $S(A, B_1 \cup B_2) = S(A, B_1) + S(A, B_2) - S(A, B_1 \cap B_2)$ (2.11)

4. $S(A, B_1 \cap B_2) = S(A, B_1)S(B_1 \cap A, B_2).$ (2.12)

Subsethood is also used to define a simpler form for entropy:

$$E(A) = S(A \cup A^C, A \cap A^C). \tag{2.13}$$

The above operators and theorems provide a framework for reasoning with fuzzy sets. With an understanding of them, we can model certain physical phenomena with fuzzy systems.

C. Fuzzy Associative Memories

Fuzzy systems describe mappings between fuzzy cubes. This provides an alternative to the propositional and predicate calculus reasoning techniques used in AI expert systems. A system designer can reason with fuzzy sets rather than propositions. The fuzzy set framework is numerical and multidimensional whereas the AI framework is symbolic and one-dimensional. Kosko explains the subtleties of this distinction as [Ref. 2: pp. 299-300]:

> *Both frameworks can encode structured knowledge in linguistic form. But the fuzzy approach translates the structured knowledge into a flexible numerical framework and processes it in a manner that resembles neural-network processing. The numerical framework also allows us to adaptively infer and modify fuzzy systems, perhaps with neural or statistical techniques, directly from problem-domain data.*

1. FAM system overview, FAM rules

Fuzzy systems are defined as mappings between fuzzy cubes. Thus fuzzy system S is a transformation, $S : I^n \rightarrow I^p$. Where I^n is defined as a unit hypercube of n-dimensions containing all of the fuzzy sets in the domain space, $\mathbf{X} = \{x_1, \ldots, x_n\}$. Similarly, I^p is defined as a unit hypercube of p-dimensions containing all of the fuzzy sets in the range space, $\mathbf{Y} = \{y_1, \ldots, y_n\}$.

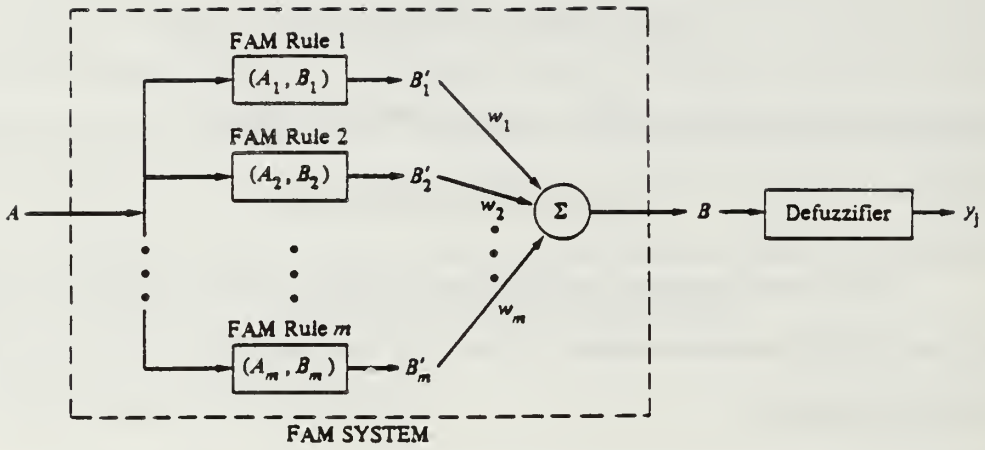

Figure 2.1: FAM System architecture [Ref. 1: p. 316].

This mapping allows the system to behave as an associative memory, mapping related inputs to corresponding outputs. This is called a fuzzy associative memory or FAM. In its simplest form, a FAM encodes a FAM rule or association (A_i, B_i) that associates p-dimensional fuzzy set B_i with an n-dimensional fuzzy set A_i.

The shape of a fuzzy set defines the membership function for that set. Although they may be of any shape, in practice fuzzy sets are defined to be trapezoidal or triangular in shape. Most systems work best when adjacent sets are defined with an overlap of approximately 25%. [Ref. 2: p. 318, p. 382]

A FAM system $F : I^n \rightarrow I^p$ encodes and processes in parallel a FAM bank of m FAM rules $(A_1, B_1), \ldots, (A_m, B_m)$ as shown in Figure 2.1. Each input A to the system activates each FAM rule to a different degree, producing a B'_i. The more A resembles A_i, the more B'_i resembles B_i. The corresponding output fuzzy set B combines these partially activated fuzzy sets $B'_1, \ldots, B'_m$. B equals the weighted average of the partially activated sets:

$$B = w_1 B'_1 + \ldots + w_m B'_m \tag{2.14}$$

where w_i reflects the credibility, frequency, or strength of the fuzzy association (A_i, B_i). In practice, the output waveform B is usually "defuzzified" to a single numerical value y_i in $\mathbf{Y}$ by computing the fuzzy centroid of B with respect to $\mathbf{Y}$.

These rules may be dictated by common sense knowledge of the general relationship between the input and output, or they can be derived adaptively from the observation of input and output training data. Kosko illustrates the adaptive generation of rules using an unsupervised procedure called product-space clustering [Ref. 2: pp. 327-335]. This procedure uses a competitive learning algorithm to position a number of vectors throughout the input-output product space. The vectors are distributed according to the density of the clusters of input-output pairs of training data. The most dense clusters attract the most vectors. Candidate rules correspond to all possible combinations of input and output fuzzy sets. These rules partition the product space into cells. The rules selected for the FAM bank are those containing the largest number of training vectors.

Determining other methods of generating these FAM rules is an ongoing research question. One method introduced recently uses genetic algorithms [Ref. 3].

FAM rules also can be compound. This allows antecedent and consequent sets to be combined with logical conjunction, disjunction, and negation operations. [Ref. 2: p. 301]

2. Fuzzy Vector-Matrix Manipulations

The association (A_i, B_i) is contained in the fuzzy $n \times p$ matrix M. The relationship is described by an operation termed max-min composition. This operation, denoted by the symbol $\circ$, is defined for row fit vectors A and B as:

$$A \circ M = B \tag{2.15}$$

where

$$b_j = \max_{1 \le i \le n} [\min(a_i, m_{ij})]. \tag{2.16}$$

The matrix M is called a fuzzy Hebb matrix and is created using correlation-minimum encoding. Elements of this matrix are defined as:

$$m_{ij} = \min(a_i, b_j), \tag{2.17}$$

and the corresponding matrix form is

$$M = A^T \circ B. \tag{2.18}$$

The above relations embody the memory characteristic of M. The vector B is recalled from M when vector A is presented to M by $A \circ M$. Further, a vector A', similar to A recalls a vector B' from M. Under certain conditions recall is bidirectional, i.e., $A \circ M = B$ and $B \circ M^T = A$.

The height, $H(A)$, of fuzzy set A is defined as the maximum fit value of A:

$$H(A) = \max_{1 \le i \le n} a_i. \tag{2.19}$$

A fuzzy set is normal if $H(A) = 1$. Recall accuracy depends on the heights $H(A)$ and $H(B)$. Normal fuzzy sets exhibit perfect recall.

An alternative to correlation-minimum is correlation-product encoding. Using this method, M is formed by the outer product of fit vectors A and B. Correlation-minimum encoding produces a matrix of clipped B sets while correlation-product encoding produces a matrix of scaled B sets.

As stated earlier, the input fit vector A is applied to each association in the FAM bank in parallel. The recalled fit vector B is "defuzzified" by combining the individual recalled vectors B'_i in a weighted sum. The scheme that is most commonly used is termed the fuzzy centroid defuzzification scheme and is given by:

$$\overline{B} = \frac{\sum_{j=1}^{p} y_j m_B(y_j)}{\sum_{j=1}^{p} m_B(y_j)}. \tag{2.20}$$

The fuzzy centroid is unique and uses all the information in the output distribution B. Computing the centroid is the only step in the FAM inference procedure that requires division. All other operations consist of inner products, pairwise minima, and additions.

3. BIOFAMs

Consider the general FAM system shown in Figure 2.1. The input vector A spans the working range of the input variable so that the elements in A represent the quantized measurements of the input variable. In practice, A is usually a unit bit vector where $x_i = 1$, and all other elements equal zero. This represents a clean input measurement. Similarly, the output vector B is usually defuzzified to a single value so that $y_j = 1$, and all other elements of B equal zero. This type of system using bit vectors for the input and output is referred to as a binary input-output FAM or BIOFAM. At this level, the system S is reduced to a mapping between Boolean hypercubes, $S : \{0,1\}^n \rightarrow \{0,1\}^p$. BIOFAMs are the most common implementation of fuzzy systems in commercial applications [Ref. 2: p. 317].

4. Correlation-minimum Inferencing

The method of determining an output vector B from an input vector A is called inferencing. The methods of inferencing most commonly used with fuzzy systems are correlation-minimum and correlation-product inferencing. This work uses correlation- minimum inferencing although either method could have been used.

This technique is illustrated here for a general compound rule $(A_p, B_q; C_r)$. In words this association would read: IF A_p AND B_q, THEN C_r. This rule represents one of the several rules used to describe the relationship between the input variables X and Y and the output variable Z.

To begin, the fuzzy system finds the membership values of the input measurements x_i and y_j in fuzzy sets A_p and B_q, which are denoted by $m^X_{A_p}(x_i)$ and $m^Y_{B_q}(y_j)$, respectively. The logical AND combination of the antecedents is performed by taking the minimum of these two values, $\text{ant} = \min(m^X_{A_p}(x_i), m^Y_{B_q}(y_j))$. The result, ant, indicates the degree to which the inputs x_i and y_j satisfy the rule $(A_p, B_q; C_r)$. The output vector is then determined by taking the pairwise minima of ant and the membership values of z_k in C_r for each element in Z. This can be expressed more concisely as

$$C'_r = \min(m^X_{A_p}(x_i), m^Y_{B_q}(y_j)) \wedge m^Z_{C_r}(z_k) = \text{ant} \wedge m^Z_{C_r}(z_k) \tag{2.21}$$

for all elements in Z where the symbol $\wedge$ denotes the minimum operation and C'_r describes a minimum scaled fuzzy set in the output space. The input measurements are presented to each of the remaining rules, and the resulting output fuzzy sets are combined by adding them pointwise as in Equation 2.14. The resulting output fuzzy set is then reduced to a single output value using Equation 2.20 to find the fuzzy centroid.

III. SPEECH CODERS

Most speech coders developed in recent years are based on linear predictive coding (LPC). This is especially true for low bit-rate speech coders. For this reason, this was the only type of speech coder considered for this work. LPC allows the broad spectral shape or spectral envelope of the speech signal to be represented by just a few parameters. Transmission of speech then consists of sending these parameters along with some representation of the finer details or residual of the signal.

A. Standard LPC

The basic idea of LPC is that the next sample of a signal can be predicted from a linear combination of several past samples. This linear combination is implemented by an FIR filter. The difference between the predicted sample and the true sample forms the residual. Because the predictable portion of the signal is removed, the residual is spectrally flatter than the original signal. Following this reasoning, if filter weights are chosen so that the predictable portion of the signal is completely removed, then the residual aproximates a white noise sequence. Conversely, if the inverse filter is driven with the white noise residual the original signal can be recovered. If the inverse filter is driven by a white noise sequence other than the residual, a signal that is statistically equivalent to the original signal is obtained.

Finding these coefficients amounts to solving the associated normal equations. There are several methods available to do this such as the Levinson recursion or the Schur algorithm. A complete discussion of LPC theory and the details of these algorithms may be found in [Ref. 4: ch. 7-8], [Ref. 5: ch. 8], and [Ref. 6].

The LPC analysis of speech begins by segmenting the speech signal into frames. The length of these frames must be short enough so that the speech signal can be assumed stationary over the duration of the frame. Typically, frame lengths are in the range of 20 ms to 40 ms. The LPC coefficients and gain for a particular frame are then determined. The remainder of the analysis procedure determines the excitation to drive the inverse or synthesis filter. The transmission packet for each frame then consists of the coefficients, gain, and the excitation or some code that would allow the excitation to be constructed. The receiver then produces the synthetic speech by forming the inverse filter with the gain and coefficients received and driving it with the indicated excitation.

Although the bit-rates achieved by LPC coders are dramatically less than PCM or ADPCM schemes, there are several drawbacks. There is an inherent delay due to the segmenting of the speech into frames. Also, this method is essentially modeling the vocal tract with an all-pole filter. While the actual speech spectrum contains zeros due to the glottal source and the vocal tract response during nasal and unvoiced sounds [Ref. 5: ch. 3]. This leads to some distortion of the signal. Further, the theory assumes that the synthesis filter is driven with a spectrally flat excitation, and the excitations used do not have a flat spectrum.

In a standard LPC speech coder, a frame is classified as either voiced or unvoiced. During voiced frames, the synthesis filter is driven with a pulse train whose pulse spacing is equal to the pitch period of the original speech. During unvoiced frames, the filter is driven with white noise. This type of speech coder produces intelligible speech of synthetic quality at bit rates of 2.4 kbit/s.

Efforts to improve the quality of the LPC speech coder have focused on improving the excitation model used. In the standard LPC coder, voiced portions are modeled as purely periodic. Actual speech contains high frequency noise components

even during strongly voiced sections. The absence of these noise components causes the synthetic speech to have a mechanical quality. Additionally, because the excitation is only of two types, it is subject to errors in making these classifications. This occurs most often during frames where the speech is transitioning from unvoiced to voiced sounds or vice versa. As a result, the synthetic speech is subject to "thumps" or bursts of noise, and odd tonal noises. Some common techniques used to construct more complex excitations are briefly described below.

B. RELP

One approach extracts perceptually important features of the residual to transmit. This is termed residual-excited linear prediction (RELP) [Ref. 5: pp. 365-370]. The residual is lowpass filtered to extract the baseband portion. The baseband portion is then decimated and the waveform is coded for transmission. The receiver then uses one of several methods to spectrally replicate this baseband signal to simulate the presence of the high frequency components.

C. Multipulse

Another approach is multipulse LPC. This approach uses an analysis-by-synthesis method to determine the optimal location and amplitude of pulses in the excitation [Ref. 7]. RELP and multipulse produce communications quality speech at bit rates in the 10-12 kbit/s range.

D. CELP

A method that has recently gained wide acceptance in industry is termed code excited linear prediction (CELP). This technique uses vector quantization to determine the excitation. An analysis-by-synthesis procedure selects the vector from a

codebook of excitation vectors that is closest to the residual in some sense. The codebook is common to both the transmitter and the receiver so only the index of the vector is transmitted. CELP was not usable for real-time applications when it was first introduced because of the computations required for the codebook search. However, with the development of fast search algorithms and advances in processor capabilities CELP has become the industry standard for mobile radio communications [Ref. 8]. CELP coders commonly produce communications quality speech at bit rates of 4.8 kbit/s [Ref. 9]. A CELP coder has also been produced that can synthesize toll quality speech at bit rates of 16 kbit/s [Ref. 10].

E. Multi-band Excitation

A technique recently introduced is termed multi-band excitation [Ref. 11]. It divides the frequency spectrum into a number of bands, typically 10-12, and assigns a voiced/unvoiced decision to each band. This allows the excitation to capture the high frequency noise components of the original speech and the harmonics of the pitch. This method originally produced high quality speech at a bit-rate of 8 kbit/s, and by applying vector quantization, the bit rate was reduced to 2.4 kbit/s [Ref. 12].

F. Mixed Excitation LPC

A final example of LPC based speech coders is the mixed-excitation LPC vocoder [Ref. 13]. This is the speech coder studied in this thesis. It uses two techniques to correct some shortcomings of the standard LPC coder. First by recognizing that voiced speech is a mixture of periodic and noise-like components, it uses an excitation made up of high pass filtered noise and lowpass filtered pulses. These are combined to provide a spectrally flat excitation where the ratio of noise standard deviation to pulse amplitude is determined by the voiced power of the speech. Secondly, the

vocoder employs a technique aimed at improving the excitation classification decision of the standard LPC coder. This is done by creating a third voicing category termed "jittery excitation." In frames classified as jittery, the pulse positions are varied by a random amount to destroy the periodicity of the excitation. This vocoder provides a dramatic improvement over the standard LPC vocoder and only costs one additional bit per frame. The quality approaches that of the 4.8 kbit/s CELP coder.

IV. DEVELOPMENT OF THE MIXED EXCITATION VOCODER

This work implementes the mixed excitation vocoder described in the previous chapter using an FAM as the excitation classifier to illustrate the use of fuzzy logic in a speech coding and classification application. The mixed excitation vocoder was chosen because it is only slightly more complex than the standard LPC vocoder yet offers a significant increase in the quality of speech produced. Additionally, the classification of speech into voiced, jittery voiced, and unvoiced classes seemed ideally suited to fuzzy logic. Another area to be explored is the use of additional excitation classes. Two or three jittery classes might better model the transition regions between voiced and unvoiced speech.

The various steps involved in the development of this work are:

- Standard LPC Vocoder: This provides a minimum performance standard for the subsequent vocoders. The MATLAB code used in this implementation provides the basis for the mixed excitation vocoders.

- Mixed Excitation (ME) Vocoder: This vocoder was written as closely as possible to the one described in [Ref. 13]. This provides another performance benchmark.

- Modified Mixed Excitation (MME) Vocoder: The vocoder described in [Ref. 13] uses autocorrelation strength and peakiness parameters to comprise the excitation classes. The MME vocoder uses energy and zero crossing rate as the decision parameters in the excitation classification. The reasons for this selection are discussed later in this chapter.

- Four Level ME Vocoder: This scheme uses additional excitation classes and examines the improvement in performance.
- Fuzzy ME Vocoder: This is the main topic of interest for this research. Versions using four and five classification levels are developed.

The remainder of this chapter discusses the development of the deterministic versions of the ME, MME, and four level ME vocoders. The fuzzy implementations are presented in the following chapter.

The simulation work is done using the SOUNDTOOL package on the Sun workstations to input and output speech files. The sampling rate of this package is fixed at 8000 samples/sec. The vocoders were simulated using MATLAB. The vocoders and the supporting routines are contained in Appendix C. The vocoders are developed in a consistent manner so that they all have the same general structure, and stand alone routines carry out low level functions. This approach made it easier to follow a logical progression from simple to progressively more complex vocoders. Placing the low level functions in stand-alone routines made it easy to experiment with various methods without changing the basic structure of the vocoder. The vocoder routines are generally arranged into two sections, the analysis implemented at the transmitter and the synthesis implemented at the receiver.

A. Standard LPC Vocoder

The analysis at the transmitter begins by sectioning the speech into frames of N samples in length. The LPC coefficients are then determined for a p^{th} order filter using the Levinson recursion. The speech signal is then filtered using the prediction error filter formed by the LPC coefficients to obtain the residual. The residual signal is used to find the pitch period. Since it is spectrally flatter than the original speech, it is easier to find the fundamental frequency from the residual. The pitch is determined

using a procedure similar to that outlined in [Ref. 14: p. 156]. The residual is first lowpass filtered using an eighth order Butterworth filter with a cutoff frequency of 600 Hz. This eliminates the effects of higher order harmonics. The signal is then center clipped to remove extraneous peaks in the autocorrelation function. The pitch period is the lag where the largest peak occurs in the normalized autocorrelation of this clipped signal. If the correlation strength of the pitch period for a given frame is greater than 0.3 and the pitch period is within a reasonable range (e.g., 0.625ms $\leq$ pitchperiod $\leq$ 15ms), the frame is voiced; otherwise it is unvoiced. A three point median filter is used to smooth the pitch period estimates. Silence regions are defined as those frames where the normalized energy of the full band residual is below a threshold of 0.002.

The transmission packet for each frame consists of the LPC coefficients, gain, and pitch period. In this implementation, a pitch period of zero denotes a silence frame, and a period of one denotes an unvoiced frame. Voiced frames are then those with pitch periods greater than one. The synthetic speech is formed frame by frame using the inverse of the filter used to obtain the residual. The inverse filter is driven by white noise scaled by the gain factor for unvoiced frames and by a pulse train scaled by the gain factor for voiced frames. The period of the pulse train is equal to the pitch period for that frame.

This vocoder produces synthetic speech that is intelligible but suffers from the distortions mentioned in the previous chapter. The distortions are particularly noticeable for female speakers and for male speakers with widely varying pitch periods. Figure 4.1 shows the correlation strengths and pitch periods for a typical male speaker where the synthetic speech suffers from these distortions. The speech has several thumps and tonals due to the binary voicing decision. Figure 4.2 shows these parameters for a typical female speaker. The synthetic speech produced in this case

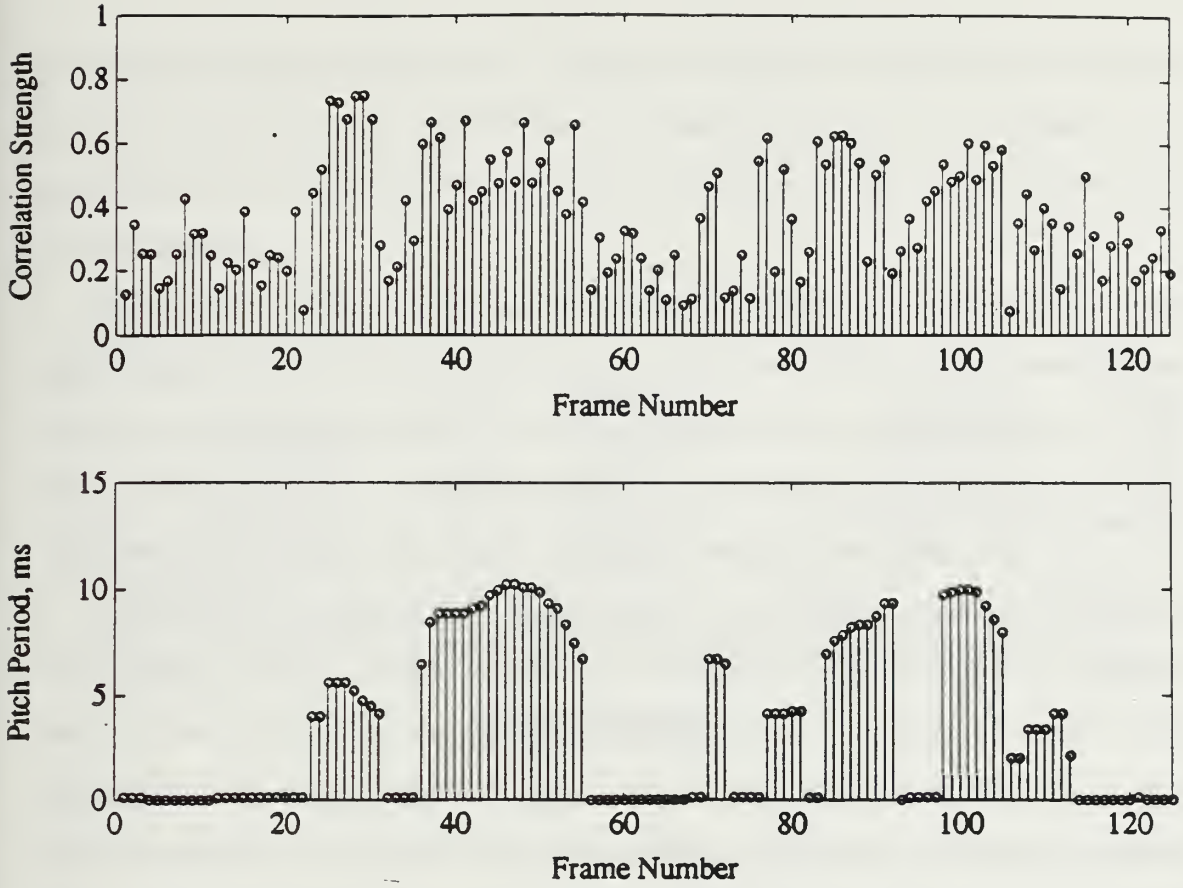

Figure 4.1: Correlation strength and pitch period from the standard LPC vocoder. The speech sample is of a male speaker saying "Excuse me madame, would you care to dance?"

is buzzy and very mechanical sounding.

B. Mixed Excitation LPC Vocoder

This vocoder is based on the work presented by McCree and Barnwell [Ref. 12]. The aim is to eliminate the buzzy or mechanical sound of the speech produced by a standard LPC vocoder. They proposed two enhancements of the LPC scheme to alleviate this distortion. The first is to use an excitation consisting of both pulse and noise excitations.

This more closely models the mixed excitation of human speech. The pulses

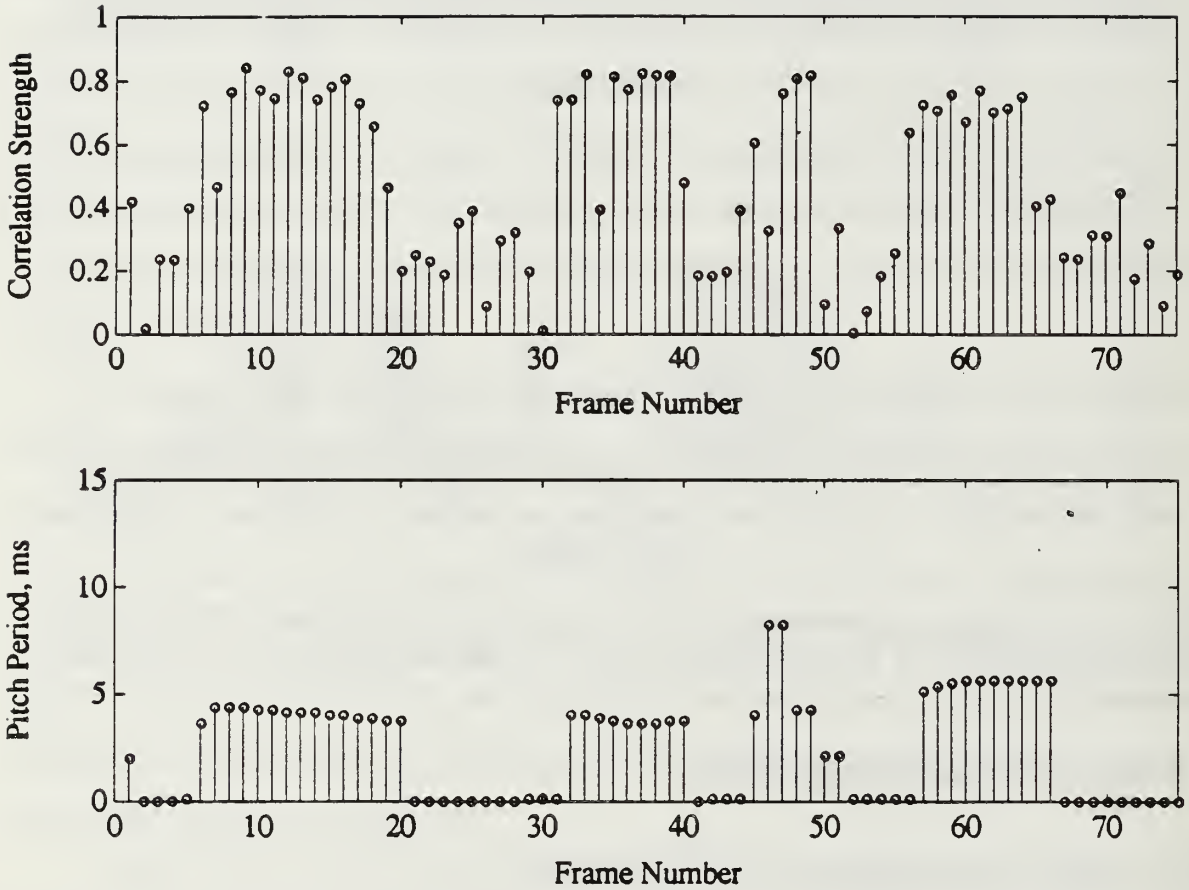

Figure 4.2: Correlation strength and pitch period from the standard LPC vocoder. The speech sample is of a female speaker saying "No, I don't think so."

are lowpass filtered and the noise is highpass filtered, so the overall excitation is spectrally flat. The mixture is based on the relative speech power of the signal and is determined at the receiver. This means that it does not require any additional bits for transmission.

The second enhancement consists of using an additional voicing class to eliminate the effects of incorrect voicing decisions. This new voicing class is termed jittery voiced and occurs most often during periods of transition from voiced to unvoiced regions. The periodicity of the pulse train for a jittery frame is perturbed by varying the pulse positions by a (uniformly distributed) random variable.

The transmitter side of this implementation is the same as that for the standard LPC implementation with the addition of the means for determining the additional voicing class. The voicing classifications are based on the correlation strength of the lowpass filtered, clipped residual signal used to find the pitch period and a parameter called peakiness. McCree and Barnwell define peakiness as the ratio of the RMS power to the average value of the full-wave rectified residual [Ref. 13]. Frames are voiced if the correlation strength is greater than 0.6, jittery voiced if peakiness is greater than 1.4 or correlation strength is between 0.2 and 0.6, and unvoiced otherwise. The transmitted parameters then consist of the LPC coefficients, gain, pitch period, and excitation classification.

The receiver side of this vocoder adds two functions to that of the standard LPC implementation: an algorithm to control the mixture of pulse and noise and a scheme to construct the jittered pulse sequences. The pulse/noise mix is determined by comparing the interpolated power of each voiced or jittery frame to an estimate of the fully voiced speech power. If the current power is within 6 dB of fully voiced, the synthetic speech is strongly voiced (80% mixture). If the current power is more than 18 dB below fully voiced, the synthetic speech is weakly voiced (50% mixture).

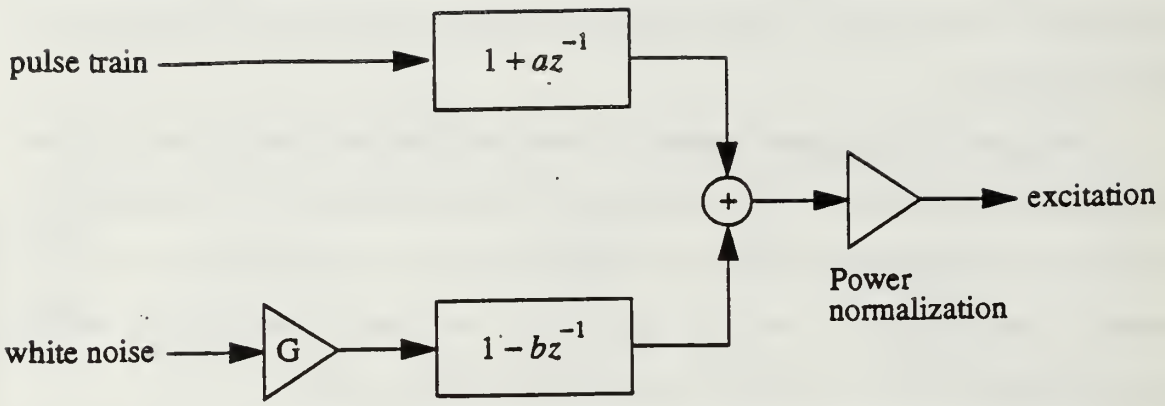

Figure 4.3: Structure for producing the mixed excitation.

For intermediate power levels, the mixture is linearly interpolated between the two values. The fully voiced power level is estimated as the maximum of the current voiced power and the previous fully voiced power estimate decayed exponentially with a time constant of 1.3 seconds.

For frames classified as jittery, the pulse positions are varied by a (uniformly distributed) random variable with value between zero and 5% of the pitch period. Pulse positions of voiced frames are varied up to 1%.

The excitation sequence is then formed by lowpass filtering the jittered pulse sequence and adding it to highpass filtered noise as shown in Figure 4.3. For this implementation, G is determined from:

$$G = 0.8 - \text{mixratio}. \tag{4.1}$$

The filter parameters are then taken to be: $a = G^2$ and $b = 1$. This structure produces excitations with spectrums that are flat over the range of mix ratios. The excitation is then used to drive the synthesis filter as before.

This model does eliminate the mechanical quality and greatly reduces the tonal

noises and thumps associated with incorrect voicing decisions. Because of this, the speech produced is clean and natural sounding. However, a new type of distortion is introduced when strongly voiced frames are classed as jittery or when the amount of jitter is excessive. When this occurs, the synthetic speech has a gargling quality. Determining the optimum amount of jitter becomes a tradeoff between the gargling distortion from too much jitter and the mechanical quality if there is not enough jitter. The amount of jitter that minmizes both distortions varies from speaker to speaker. Although the female speakerin our tests required less jitter, one of the male speakers required more jitter. The values chosen above provide reasonable performance for a wide range of speakers.

This model was tested using several speakers and with various numbers of coefficients and frame lengths. The performance was consistent from speaker to speaker. Increasing the number of coefficients used in the LPC analysis improved the quality of the synthetic speech. There is a dramatic improvement in increasing from eight to ten coefficients and only a slight improvement in going from ten to sixteen coefficients. Ten coefficients seemed to provide a good balance between speech quality and computational complexity. Frame length was tested at 18.5 ms, 25 ms, and 37.5 ms. As expected, the shorter frame lengths perform best. Frame lengths less than 18.5 ms caused problems in finding the pitch and the quality of speech produced using the 37.5 ms frames was quite poor due to the signal being non-stationary over the analysis frame. This is consistent with the rule of thumb that analysis frames should not exceed 30ms. Based on the results above, the author decided to use 10 coefficients and 25 ms windows as typical values in subsequent work. Figure 4.4 and Figure 4.5 show the decision parameters, pitch periods, and excitation classes for a male and a female speaker using these typical values.

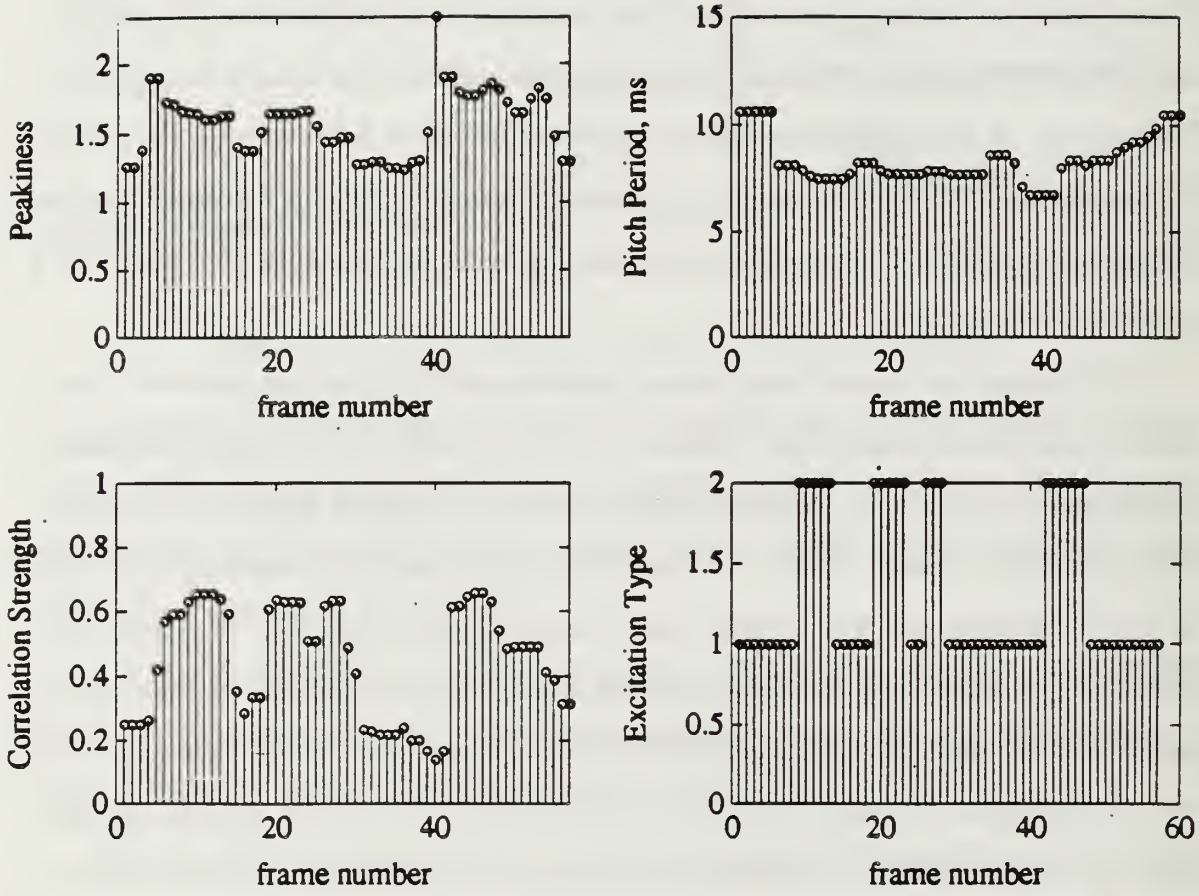

Figure 4.4: Peakiness, correlation strength, pitch period, and excitation type from the Mixed Excitation vocoder. The speech sample is of a male speaker saying "Asian cattle." The excitation types are UV for 0, JV for 1, and VO for 2.

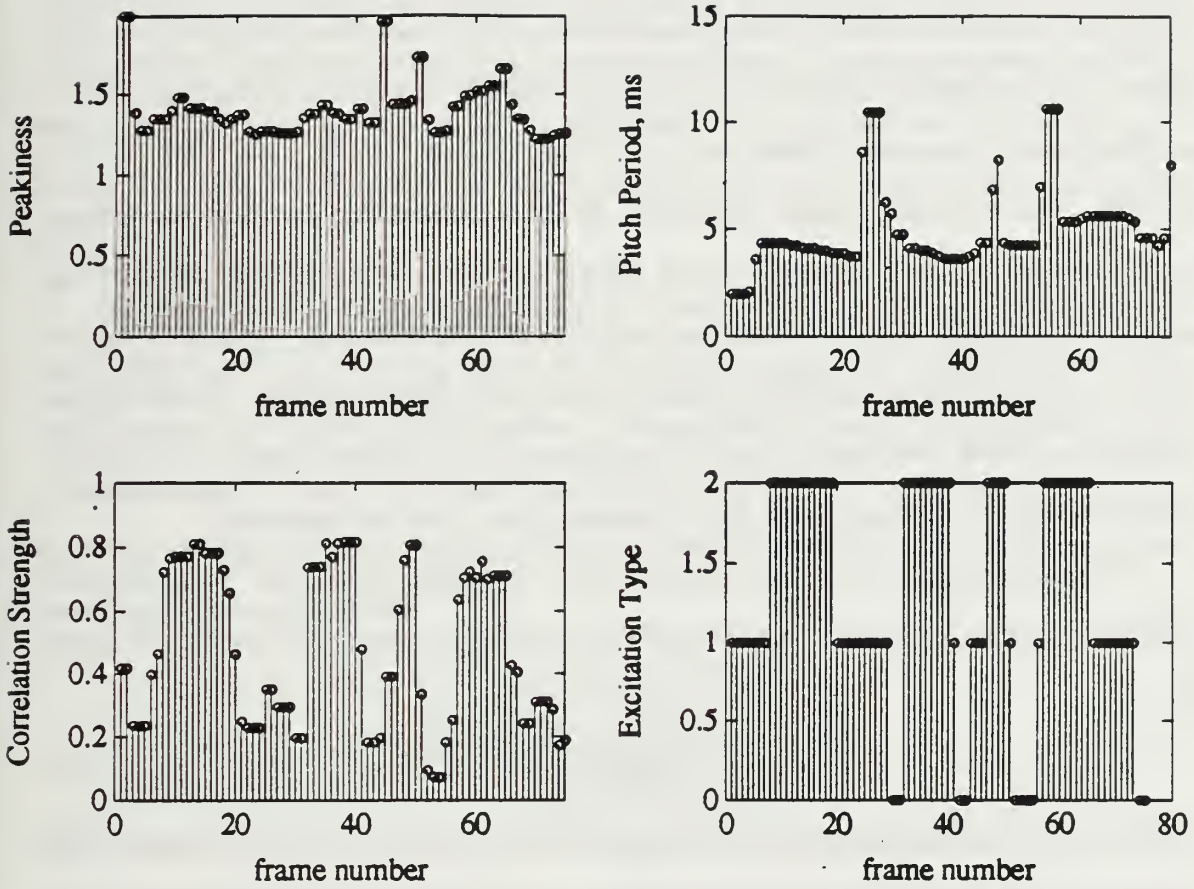

Figure 4.5: Peakiness, correlation strength, pitch period, and excitation type from the Mixed Excitation vocoder. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV for 1, and VO for 2.

C. Modified Mixed Excitation Vocoder

The mixed excitation vocoder described above uses two parameters to classify the excitation of a frame; however, the peakiness actually contributes little to the decision. For the fuzzy implementation it was desired to use two decision parameters that would contribute equally to the classification. Two quantities that meet this criterion, which are easily computed are short time energy and zero crossing rate. These are often used to separate silence and voiced regions [Ref. 14: p. 130], [Ref. 5: pp. 213-215]. The reasoning behind the choice of these parameters was that the energy would be high during periods of voiced speech and low during periods of unvoiced speech. Conversely, the zero crossing rate would be low during periods of voiced speech and high during the noisy regions of unvoiced speech. Before discussing the implementation of these parameters to determine the excitation class, the impact of windowing effects on these short time functions must first be considered.

By sectioning the data into frames with N samples per frame, a rectangular window is applied by default. The impulse response of a rectangular window is given by:

$$H(e^{j\Omega T}) = \frac{\sin(\Omega NT/2)}{\sin(\Omega T/2)} e^{-j\Omega T(N-1)/2}, \tag{4.2}$$

where T is the sampling period in seconds and $\Omega = 2\pi f$. The first zero of this response occurs at the analog frequency of $f = f_s/N$ where $f_s = 1/T$ is the sampling frequency. This is normally taken to be the cutoff frequency so that the bandwidth of a rectangular window is

$$BW_r = \frac{1}{NT}. \tag{4.3}$$

These short-time quantities are essentially a result of convolution of the transformed speech signal and the window [Ref. 5: p. 212], i.e., the short-time quantity being

computed is given by:

$$Q(n) = \sum_{m=-\infty}^{\infty} T[s(m)]w(n-m) \tag{4.4}$$

where the transformation $T[\cdot]$ would amount to a squaring operation for energy. To compute $Q(n)$ accurately, it must be calculated at a rate of at least $2BW_r$, or every $N/2$ samples. This means that $Q(n)$ is computed using N sample windows with 50% overlap. Therefore, while the LPC coefficients are computed once per frame, the energy and zero crossing rate are computed twice in each frame or once every half frame.

To implement the ME vocoders using these parameters, the normalized energy and normalized zero crossing rates are computed using the original speech. The pitch periods are determined as before. The excitation classes are modified so that a classification is made every half frame. The receiver is also modified so that the excitation can be updated every half frame as well. The excitation decisions are made using the following matrix:

Energy \ Zero Crossing Rate			
	UV	*JV*	*UV*
	VO	*JV*	*JV*
	VO	*VO*	*JV*

(4.5)

The thresholds were adjusted so that the performance was close to that of the vocoder using correlation strength and peakiness. This resulted in thresholds of 0.15 and 0.6 for the zero crossing rate; 0.05 and 0.6 for the energy.

There are several problem areas with this implementation. One of these is that the synthetic speech does not reproduce plosives well. This problem is also shared by the implementation using the correlation strength and peakiness parameters.

In addition, the MME implementation sometimes classifies nasals as unvoiced causing the synthetic speech to have a noisy quality during these sounds. This problem could be reduced by adjusting the threshold between low energy and moderate

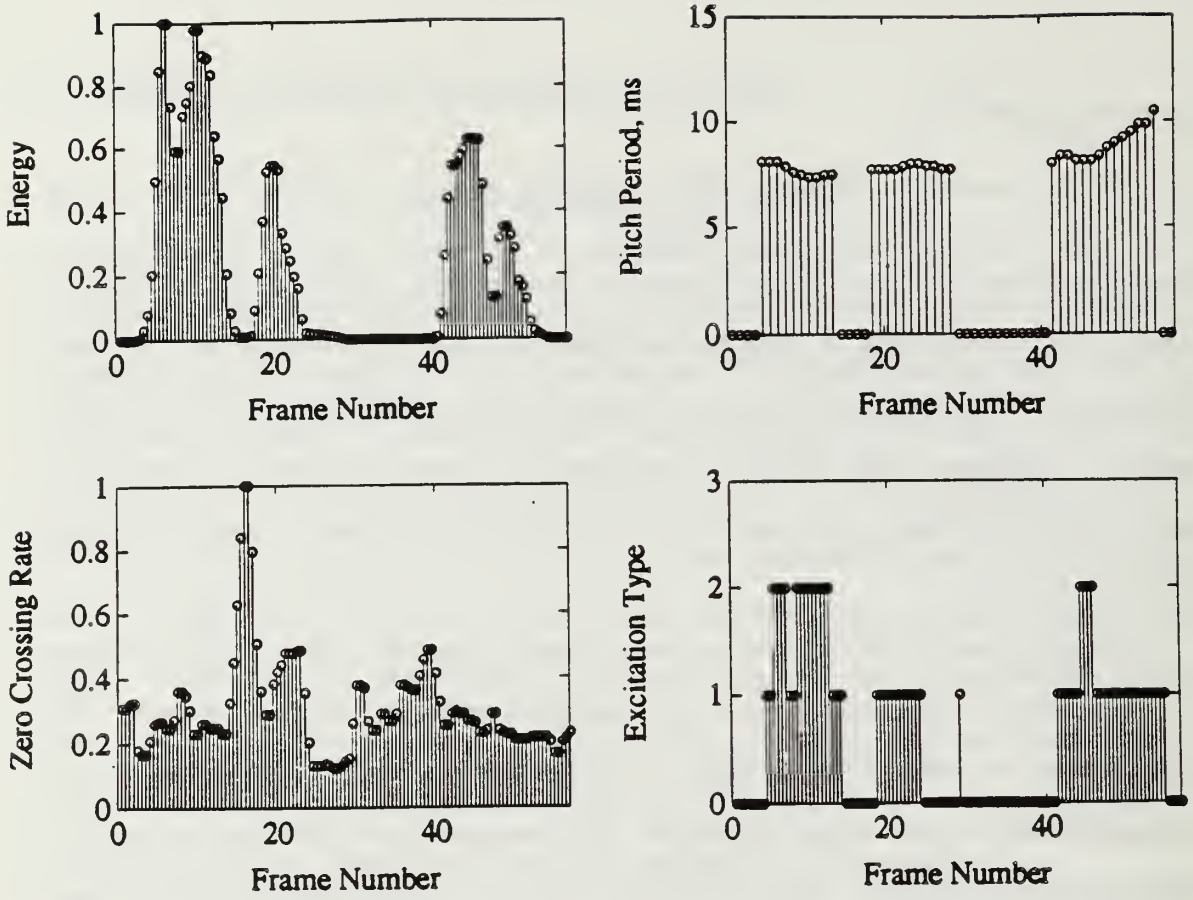

Figure 4.6: Energy, zero crossing rate, pitch period, and excitation type from the MME vocoder. The speech sample is of a male speaker saying "Asian cattle." The excitation types are UV for 0, JV for 1, and VO for 2.

energy, but doing so would have an adverse impact on other sounds. An example where this occurs is shown in Figure 4.6. The phrase spoken is, "Asian cattle". The "n" in "Asian" occurs between frames eighteen and twenty-eight. Approximately half of these frames are classified as unvoiced.

A final problem noted is that, in some cases, not all of the excitation classes are utilized. Figure 4.7 is an example of this. It uses the same original speech as in Figure 4.5, but the classes were quite different. This problem is alleviated somewhat with the addition of an additional excitation type. However, it does indicate that the thresholds vary from speaker to speaker and must be adjusted for optimum performance.

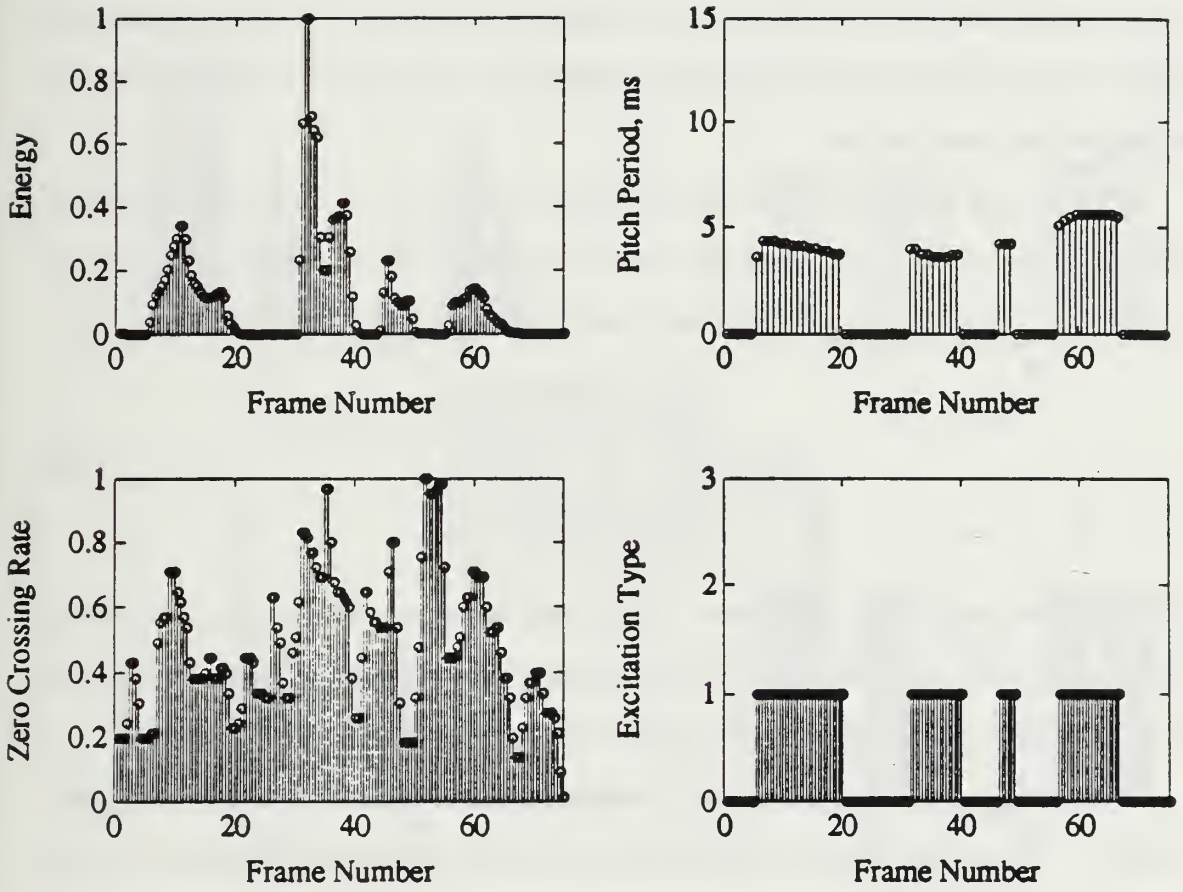

Figure 4.7: Energy, zero crossing rate, pitch period, and excitation type from the MME vocoder. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV for 1, and VO for 2.

D. Four Level MME Vocoder

The reasoning for creating a fourth excitation level is to provide a smoother transition between voiced and unvoiced regions. Just as the level of noise increases as the speech transitions from voiced to unvoiced, the pitch periods become more erratic. Also, this fourth level does not require additional bits in the transmission packet. Since two bits were being used to specify three levels, the fourth level would better utilize these two bits.

The voicing classes for this implementation were UV, J1, J2, and VO. Pulse positions were moved up to 5% of the pitch period for J1, up to 3% for J2, and up to 1% for VO. The classes are determined according to the following decision matrix:

	Zero Crossing Rate			
Energy	*UV*	*J2*	*J1*	*UV*
	J2	*J1*	*J1*	*J1*
	VO	*J2*	*J2*	*J1*
	VO	*VO*	*J2*	*J1*

(4.6)

The extra excitation class was formed by splitting the most populous excitation class of the three level classifier. The new thresholds used were 0.15, 0.25, and 0.6 for normalized zero crossing rate; 0.05, 0.15, and 0.55 for normalized energy.

The performance of the four level classifier is better than that of the three level classifier. Comparing Figure 4.8 to Figure 4.6 shows how the additional class was used. The problem noted above with the female speech sample was also alleviated somewhat as seen in Figure 4.9.

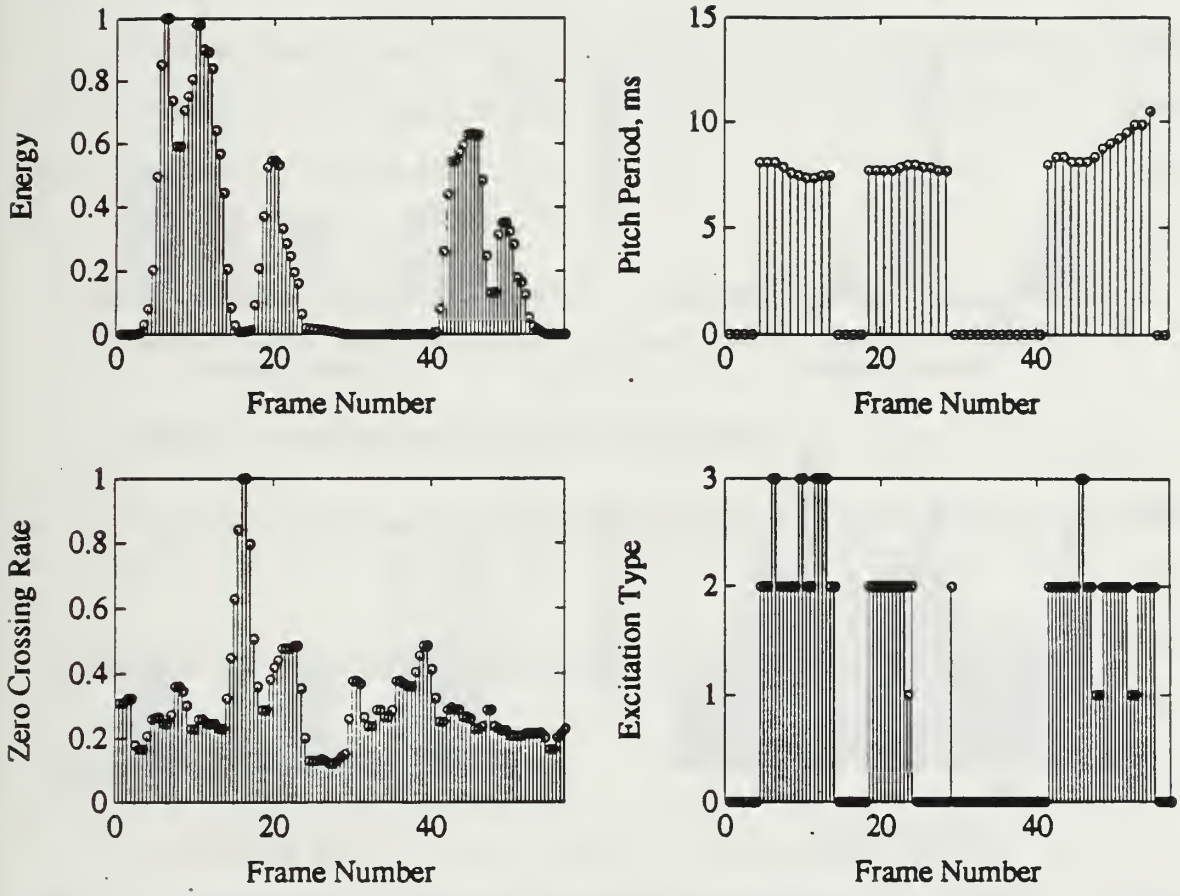

Figure 4.8: Energy, zero crossing rate, pitch period, and excitation type from the four level MME vocoder. The speech sample is of a male speaker saying "Asian cattle." The excitation types are UV for 0, JV1 for 1, JV2 for 2, and VO for 3.

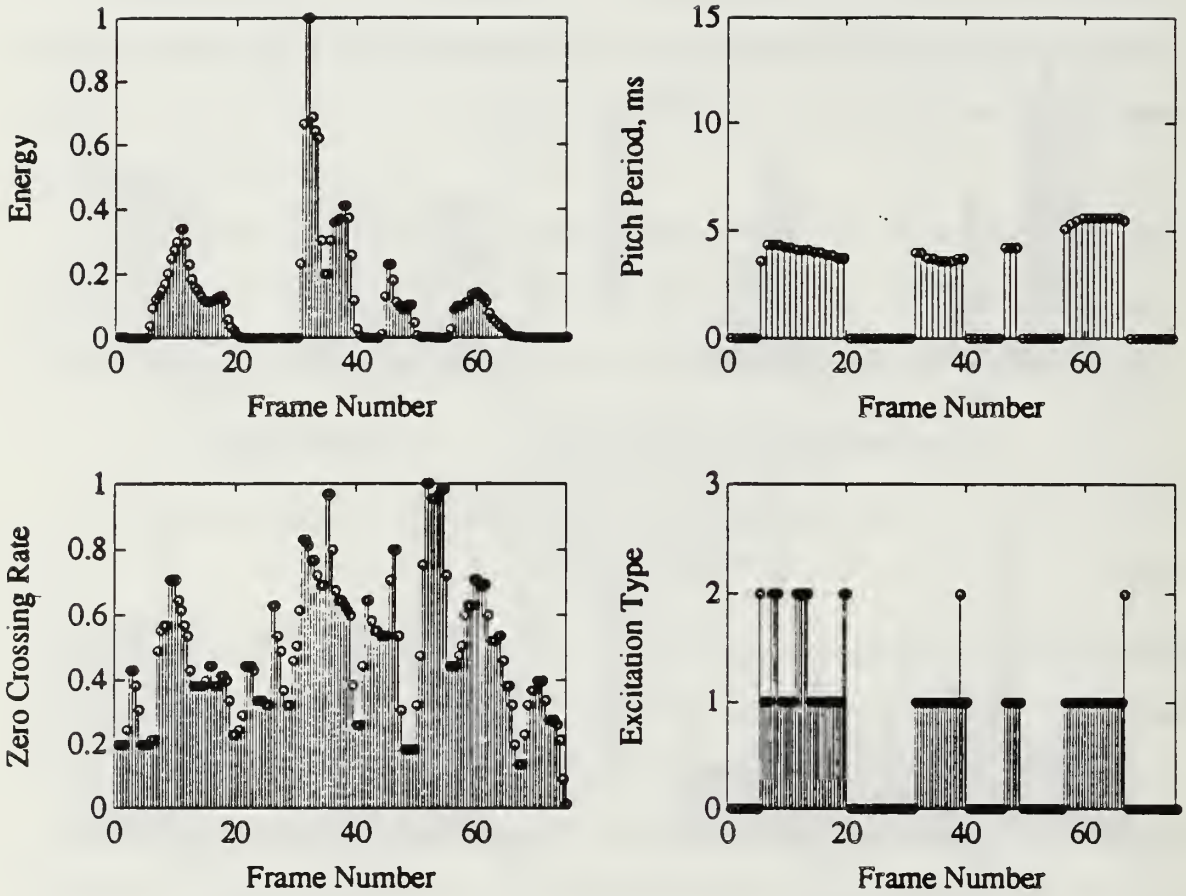

Figure 4.9: Energy, zero crossing rate, pitch period, and excitation type from the four level MME vocoder. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV1 for 1, JV2 for 2, and VO for 3.

V. IMPLEMENTATION OF THE FUZZY EXCITATION CLASSIFIER

The final phase of development in this work involves implementing the MME vocoder using a FAM to obtain the excitation classes. The design of this fuzzy system proceeded in four broad steps:

- Define the fuzzy variables.
- Define the shape and boundaries of fuzzy sets for each fuzzy variable.
- Define the associations between these fuzzy sets.
- Place the selected associations in a FAM bank structure to perform the classification.

These steps are discussed in detail below.

A. The Fuzzy Variables

The variables are the same as those in the deterministic vocoder. For this implementation the input variables are normalized energy and normalized zero crossing rate, as previously discussed. The output variable is the excitation classes: unvoiced, one or more classes of jittery voiced, and fully voiced.

B. Finding the Fuzzy Sets

Determining the optimum size and shape of fuzzy sets is an ongoing research topic in the development of fuzzy systems. The sets are typically made triangular

or trapezoidal in shape for simplicity. Kosko presents a case study which shows that the optimal overlap between adjacent sets in a control application is approximately 25% [Ref. 2: ch. 11]. However, he does not suggest a method of determining how many sets one should use or how large each set should be. Genetic algorithms have been used in the literature [Ref. 3] to design fuzzy sets and to determine the optimum associations.

The fuzzy sets in this implementation are triangular or trapezoidal in shape. The author decided to keep the number of sets equal for each variable. This was not necessary but was done to maintain the symmetry of the inverse relationship between energy and zero crossing rate. The location of the set boundaries are found by forcing the sets to conform to a distribution. For a given number of fuzzy sets of one variable, x percentage of the variable samples are made to fall within the first fuzzy set, y percent within the second set, and so on. The weights used in this work were determined by examining speech samples taken from four different speakers. The weights were then chosen to give a good distribution of excitation classes regardless of the speaker. Obviously these weights could be refined using more sophisticated analysis techniques and using a larger number of speakers. For energy, the weights from low energy to high energy are: 0.40, 0.19, 0.19, and 0.22 for the four level classifier; 0.40, 0.15, 0.15, 0.15, and 0.15 for the five lever classifier. The weights for zero crossing rate were from low to high: 0.06, 0.40, 0.36, and 0.18 for the four level; 0.05, 0.26, 0.26, 0.26, and 0.17 for the five level.

In this implementation, the fuzzy sets are determined for each speech file that is processed. In a real time application, the fuzzy sets could easily be made adaptive. The set boundaries would update continuously based on the recent history of the speech processed.

C. Defining the Associations

Decision matrices were derived using an intuitive approach and an empirical approach. This illustrates one of the most appealing aspects of fuzzy systems. A system can be put together quickly using the designer's intuitive understanding of the problem, or the associations can be derived empirically using training data and some type of learning algorithm. This allows a prototype to be developed quickly and the learning algorithms can then be used to refine the design or to gain new insight into the problem. Both methods used are outlined in the following subsections.

1. Intuitive Decision Matrix

Since there are two input variables, the associations are compound with the input variables combined with an AND function. One association could be articulated as: IF the energy is HIGH AND the zero crossing rate is LOW, THEN the speech is VOICED. For the four level classifier, there would then be sixty-four possible associations. Of these, sixteen are required to cover all combinations of the input variables. These sixteen associations can then be inferred from the inverse relationship between energy and zero crossing rate as shown above. This is how the decision matrix for the deterministic vocoder was derived. Based on this the decision matrix used was:

		Zero Crossing Rate			
		$Z1$	$Z2$	$Z3$	$Z4$
Energy	$E1$	UV	$J2$	$J1$	UV
	$E2$	$J2$	$J1$	$J1$	$J1$
	$E3$	VO	$J2$	$J2$	$J1$
	$E4$	VO	VO	$J2$	$J1$

(5.1)

2. Empirically Derived Decision Matrix

The empirical matrix was derived using an offline method that analyzed almost 35 seconds of speech data from four different speakers. As shown in Equation 5.1, the matrix must contain sixteen rules in order to cover all possible input combinations. Each cell in the matrix has four possible consequents. The method begins by determining which of the sixteen rules are best applied to a given energy and zero crossing rate input pair. Next, it produces candidate excitations using all four of the excitation types. It then uses a distance measure to find the best candidate excitation when compared with the residual. The distance measure used was the same type used in CELP speech coders as outlined in [Ref. 8]. Once the winning candidate is known, the appropriate entry in a tally matrix is incremented. The tally matrix is a 16×4 matrix containing the number of times each type of excitation won for each rule. The maximum entry in each row then determines the consequent for each rule. This method produced the following matrix using speech samples from four different speakers:

		Zero Crossing Rate			
		$Z1$	$Z2$	$Z3$	$Z4$
Energy	$E1$	$J2$	VO	$J2$	VO
	$E2$	$J1$	VO	UV	$J2$
	$E3$	$J2$	$J1$	UV	$J2$
	$E4$	VO	UV	UV	$J1$

(5.2)

The entries here are quite different from those in the matrix of Equation 5.1 generated by intuition. The reason for this is that no clear trend emerged in the cumulative totals. In nine of the sixteen rules, the winning consequent was less than 8% larger than the next larger consequent. The largest winning margin was only 22%. Possible explanations for the poor results are:

- There is not enough distinction between one excitation type and another.

- The rule matrix is speaker dependent.

The author feels that the first possibility is the major reason but cannot be avoided without using a more complex vocoder model. The second possibility also has some validity and is easily tested. The above method is used to produce matrices for each individual speaker. They are all quite different from each other and different from Equation 5.2. One of these individual matrices is included here for illustration:

		Zero Crossing Rate			
		$Z1$	$Z2$	$Z3$	$Z4$
Energy	$E1$	$J2$	UV	VO	VO
	$E2$	VO	VO	VO	$J2$
	$E3$	VO	$J1$	VO	VO
	$E4$	VO	UV	$J2$	UV

(5.3)

Similar to the cumulative matrix, these individual matrices show no real trend in the winning candidates. This lends more credibility to the first possibility above.

D. Implementing the FAM Structure

Once the fuzzy sets are defined and the associations are selected, the implementation is fairly straightforward. The classifications are made once every half frame. The vocoder presents the new values of energy and zero crossing rate to each association in the FAM bank. The degree to which the new data satisfies a given association, say (E4,Z1;VO), is determined using the correlation-minimum inferencing technique. Once all of the associations have been poled, the output is "defuzzified" to a single value using Equation 2.5. This value is then rounded to the nearest classification level. The remainder of the vocoder is the same as in the deterministic four level implementation.

E. Five Level Fuzzy Classifier

A five level fuzzy classifier was also implemented using an intuitive rule matrix. The amount of jitter is distributed as follows: $J1 = 4\%$; $J2 = 3\%$; $J3 = 2\%$; and $VO = 1\%$. The decision matrix used was developed in the same manner as the four level matrix of Equation 5.1 and is listed here:

		Zero Crossing Rate				
		$Z1$	$Z2$	$Z3$	$Z4$	$Z5$
Energy	$E1$	UV	$J3$	$J2$	$J1$	UV
	$E2$	$J2$	$J2$	$J1$	$J1$	$J1$
	$E3$	$J3$	$J3$	$J2$	$J2$	$J1$
	$E4$	VO	$J3$	$J3$	$J2$	$J2$
	$E5$	VO	VO	$J3$	$J2$	$J2$

(5.4)

F. Results

The four level fuzzy implementation using the matrix of Equation 5.1 performed better than the four level deterministic model. As seen in Figure 5.1 and Figure 5.2, the classifier uses all of the classifications available. This is due to the way the fuzzy sets are defined; i.e., the fuzzy sets are defined for each individual speaker. Figure 5.1 also shows that the nasal "n" from frames 18–28 are now correctly classified. The five level implementation offers a slight improvement over the four level model.

Speech produced using the implementation made with the empirical matrix of Equation 5.2 is not of the same quality as that produced using the intuitive matrix. The reasons for this are discussed above. Using the matrices of the individual speakers as in Equation 5.3 improved the speech quality slightly over that of the cumulative empirical implementation.

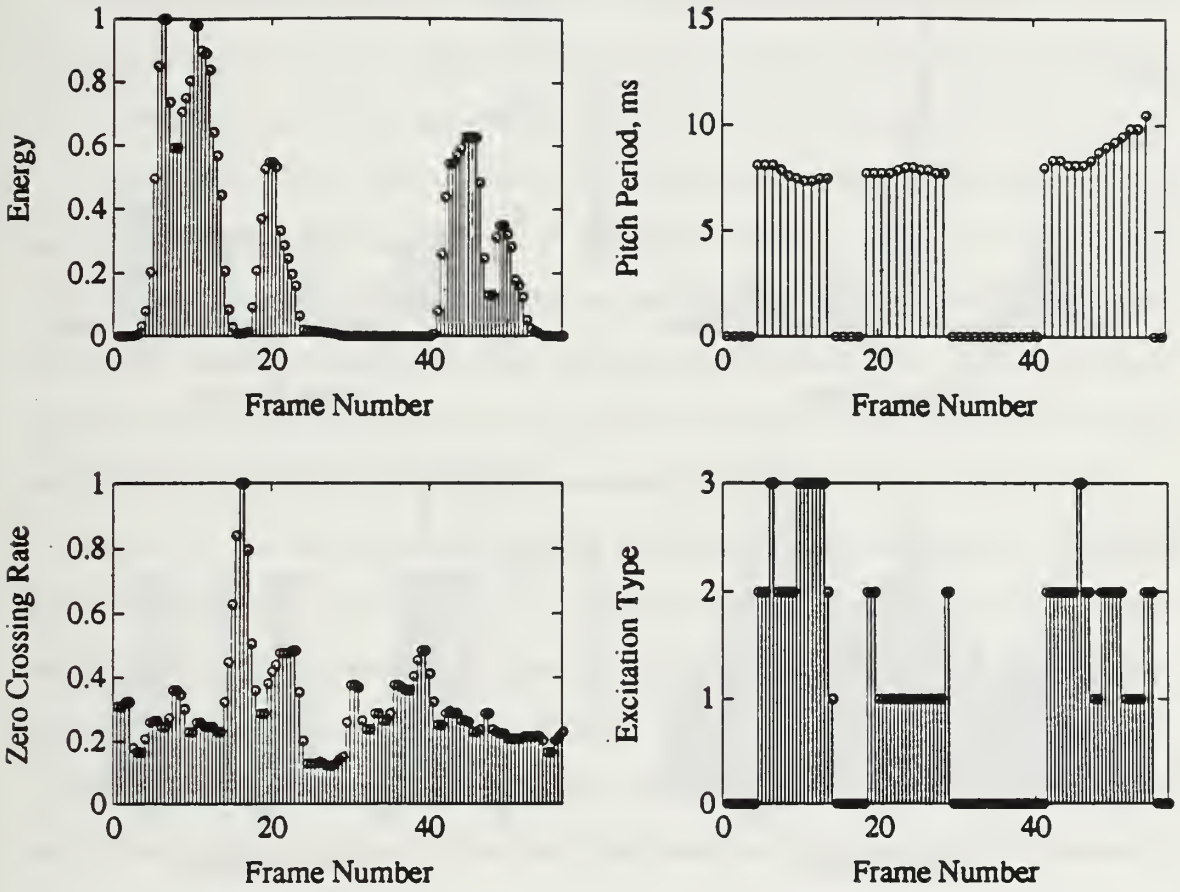

Figure 5.1: Energy, zero crossing rate, pitch period, and excitation type from the four level fuzzy MME vocoder. The speech sample is of a male speaker saying "Asian cattle." The excitation types are UV for 0, JV1 for 1, JV2 for 2, and VO for 3.

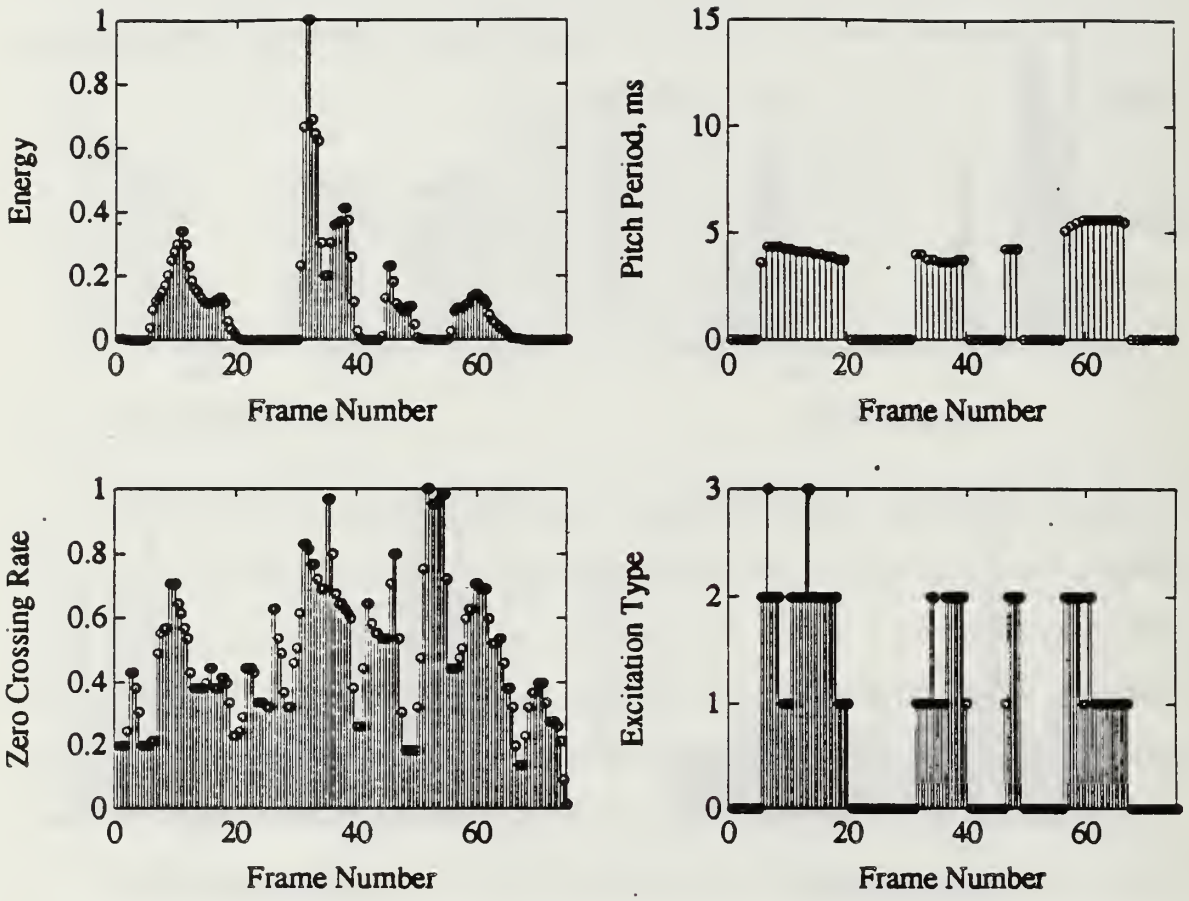

Figure 5.2: Energy, zero crossing rate, pitch period, and excitation type from the four level fuzzy MME vocoder. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV1 for 1, JV2 for 2, and VO for 3.

VI. CONCLUSIONS

This work demonstrates that a fuzzy system can perform quite well in a speech coding application. The principles used here could easily be extended to more complex classification problems and other areas of speech and signal processing.

The thesis develops a mixed excitation vocoder using normalized values of energy and zero crossing rate as parameters to determine the type of excitation to be used. This vocoder is termed the modified mixed excitation or MME vocoder to distiguish it from the mixed excitation vocoder developed in [Ref. 13]. The mixed excitation vocoder greatly improves the quality of speech produced by a standard LPC vocoder with only a modest increase in the transmission bit rate.

Four and five level implementations of the MME are developed. Additional excitation levels with varying amounts of pulse position jitter result in some performance improvement. The improvement in performance is most noticable in transition regions between voiced and unvoiced speech.

The thesis presents implementations of the MME using a fuzzy logic based excitation classifier. The use of the fuzzy logic based excitation classifier improves the performance of the MME. This is because the classifier adapts to the characteristics of each individual speaker. An implementation of the fuzzy logic based excitation classifier using an empirically derived rule matrix is presented. This implementation does not perform as well as the implementation using an intuitively derived rule matrix. The reason for this is that the distinction between one excitation class and another is not great enough to provide a clear trend for a learning algorithm.

Additional work with this type of vocoder should focus on constructing a more complex excitation. Such an excitation could be produced by applying the classifi-

cation process to several frequency bands. This would be similar to the multi-band approach of [Ref. 11]. Improving the excitation has the greatest potential for improving the vocoder performance.

This work shows that fuzzy logic is well suited for the task of designing adaptive classifiers. Further work could explore other possible applications of fuzzy logic such as radar and dynamic noise rejection. Research opportunities also exist in the areas of devising methods to define fuzzy sets within a system and in training a system to learn associations adaptively.

APPENDIX A
MATLAB ROUTINES

This appendix contains the MATLAB routines developed in support of the thesis. They are arranged according to the type of funtion they perform.

A. Speech Coders

Two examples have been included. The other implementations follow a similar format.

- ME4L.m: Four level MME
- fuzME.m: Fuzzy four level MME
- rcanal4L.m: receiver function

B. Speech Analysis

- rwind.m: Applies a rectangular window to a speech file. Sections the data into frames of length N.
- stcorr.m: Performs the short-time correlation function.
- levinson.m: Performs the Levinson recursion on p frames of speech data.
- txanal.m: Obtains the residual from the speech signal.
- Lpass.m: Used to low pass filter the residual prior to finding the pitch periods.
- clip.m: Center clips the input signal.

- pitchper.m: Estimates the pitch period for each frame of input data.
- medfilt.m: Applies a median filter of specified length to the given data.
- enrgy.m: Computes the energy with 50% overlap of frames.
- zerocrs.m: Computes the zero crossing rate with a 50% overlap of frames.

C. Fuzzy Logic Analysis

- mkset.m: Defines a trapezoidal fuzzy set when given the desired breakpoints.
- mship.m: Determines the membership of the value x in the fuzzy set A.
- corminFAM.m: Performs the correlation-minimum inferencing procedure.
- findset.m: Defines a specifed number of fuzzy sets for the input data using a distribution contained in the input weighting vector.

```
ses ZCR and ENERGY as decision parameters; 4 voicing classes.
nfile = 'ME4L.m';

lename = input('name of sound file to process?','s');
 = input('frame length?');
 = 200;
 = input('LPC filter order?');
 = 10;
 = 8000;

ta = getsound(filename);
ndow = rwind(data,N);
,numframes] = size(window);
mm = (hamming(N)*ones(1,numframes)).*window;    %apply hamming window to each frame;
 = stcorr(hamm,p);
sp('finding LPC coefficients')
sigma, a ,gamma] = levinson(R,p);
C = [sqrt(sigma.') a.'];
sp('getting residual')
s = txanal(LPC,window);

sp('energy')
g = enrgy(window);
g = medfilt(eng,3);                          %apply 3-pt median filter
g = eng./max(eng);

sp('Zero crossings');
cr = zerocrs(window);
cr = medfilt(zcr,3);                         %apply 3-pt median filter
cr = zcr./max(zcr);

res = Lpass(res);                              % low pass filter residual
Lres = clip(Lres,N);                           %clip the signal

lRres = stcorr(cLres,N-1);                     %correlate clipped signal
lRres = clRres.*(ones(N,1)*(1.0 ./ clRres(1,:)));

sp = stcorr(window,N-1);                     %correlate speech signal
sp = Rsp.*(ones(N,1)*(1.0 ./ Rsp(1,:)));

sp('finding pitch periods')
p = pitchper(clRres,Rsp);

Determine Speech Class for each half-frame
V=0; J1= 1; J2 = 2; VO = 3;
sp('determining speech classes')
thresL = 0.05;
thresM = 0.15;
thresH = 0.55;
thresL = 0.15;
thresM = 0.25;
thresH = 0.6;
lass = zeros(2*numframes,1);
or i = 1:2*numframes
  if (eng(i) <= ethresL)
     if (zcr(i) > zthresL)
```

```
         if(zcr(i) <= zthresM)
            class(i) = J2;
         elseif (zcr(i) <= zthresH)
            class(i) = J1;
         end
      end
   else
      if (eng(i) <= ethresM)
         if (zcr(i) <= zthresL)
            class(i) = J2;
         else
            class(i) = J1;
         end
      elseif (eng(i) <= ethresH & eng(i) > ethresM)
          if (zcr(i) <= zthresL)
             class(i) = V0;
          elseif (zcr(i) <= zthresH)
             class(i) = J2;
          else
             class(i) = J1;
          end
      else
         if (zcr(i) <= zthresM)
            class(i) = V0;
         elseif (zcr(i) <= zthresH)
            class(i) = J2;
         else
            class(i) = J1;
         end
      end
   end
end

LPC = [LPC pp];
class = reshape(class,2,numframes)';
nopitch = find(pp == 0);
class(nopitch,:) = zeros(length(nopitch),2);

save ME4L

% The Transmit side is everything down to this point. %%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% From here down is the Receiver side %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

shatN = rcanal4L(LPC,N,class,fs,filename,runfile);
pfileN = ['ME4L' filename num2str(zonk)];
disp('storing output file')
putsound(shatN,pfileN);

save ME4L
```

```
Uses ZCR and ENERGY as decision parameters; 4 voicing classes.
unfile = 'fuzME.m';

filename = input('name of sound file to process?','s');
 = 200;
 = 10;
s = 8000;

ata = getsound(filename);
indow = rwind(data,N);
N,numframes] = size(window);
amm = (hamming(N)*ones(1,numframes)).*window; %apply hamming window to each frame
 = stcorr(hamm,p);
isp('finding LPC coefficients')
sigma, a ,gamma] = levinson(R,p);
PC = [sqrt(sigma.') a.'];
isp('getting residual')
es = txanal(LPC,window);


isp('making fuzzy sets')
ng = enrgy(window);
ng = medfilt(eng,3);                          %apply 3-pt median filter
ng = eng./max(eng);
ngsets = findsets(eng,[.40,.19,.19,.22],4);
cr = zerocrs(window);
cr = medfilt(zcr,3);                          %apply 3-pt median filter
cr = zcr./max(zcr);
crsets = findsets(zcr,[.06,.40,.36,.18],4);
1 = engsets(1:4,:);E2 = engsets(5:8,:);E3 = engsets(9:12,:);E4 = engsets(13:16,:);
1 = zcrsets(1:4,:);Z2 = zcrsets(5:8,:);Z3 = zcrsets(9:12,:);Z4 = zcrsets(13:16,:);

classification sets (output)
V  = mkset(0,0,0, 0.7);
1 = mkset(0.3 ,1.0,1.0,1.7);
2 = mkset(1.3 ,2.0 ,2.0 ,2.7);
O  = mkset(2.3,3.0 ,3.5 ,3.5 );



res = Lpass(res);                             % low pass filter residual
Lres = clip(Lres,N);                          %clip the signal

lRres = stcorr(cLres,N-1);                    %correlate clipped signal
lRres = clRres.*(ones(N,1)*(1.0 ./ clRres(1,:)));

Rsp = stcorr(window,N-1);                         %correlate speech signal
Rsp = Rsp.*(ones(N,1)*(1.0 ./ Rsp(1,:)));

disp('finding pitch periods')
p = pitchper(clRres,Rsp);

Determine speech class for each half-frame
class = zeros(2*numframes,1);                       %zeros will mean unvoiced
disp('determining speech classes');
```

```
save fuzME
rulecnt = zeros(16,1);

for i = 1:2*numframes
classrng = [0.0:0.1:3.5];
fuzout = zeros(classrng);

%%%%% FAM BANK %%%%%
temptot= corminFAM([mship(eng(i),E1) mship(zcr(i),Z1)],UV,classrng); %(E1,Z1;UV)
fuzout = fuzout + temptot;
rulecnt(1) = rulecnt(1) + sum(temptot);

temptot= corminFAM([mship(eng(i),E2) mship(zcr(i),Z1)],J2,classrng); %(E2,Z1;J2)
fuzout = fuzout + temptot;
rulecnt(2) = rulecnt(2) + sum(temptot);

temptot= corminFAM([mship(eng(i),E3) mship(zcr(i),Z1)],VO,classrng); %(E3,Z1;VO)
fuzout = fuzout + temptot;
rulecnt(3) = rulecnt(3) + sum(temptot);

temptot= corminFAM([mship(eng(i),E4) mship(zcr(i),Z1)],VO,classrng); %(E4,Z1;VO)
fuzout = fuzout + temptot;
rulecnt(4) = rulecnt(4) + sum(temptot);

temptot= corminFAM([mship(eng(i),E1) mship(zcr(i),Z2)],J2,classrng); %(E1,Z2;J2)
fuzout = fuzout + temptot;
rulecnt(5) = rulecnt(5) + sum(temptot);

temptot= corminFAM([mship(eng(i),E2) mship(zcr(i),Z2)],J1,classrng); %(E2,Z2;J1)
fuzout = fuzout + temptot;
rulecnt(6) = rulecnt(6) + sum(temptot);

temptot= corminFAM([mship(eng(i),E3) mship(zcr(i),Z2)],J2,classrng); %(E3,Z2;J2)
fuzout = fuzout + temptot;
rulecnt(7) = rulecnt(7) + sum(temptot);

temptot= corminFAM([mship(eng(i),E4) mship(zcr(i),Z2)],VO,classrng); %(E4,Z2;VO)
fuzout = fuzout + temptot;
rulecnt(8) = rulecnt(8) + sum(temptot);

temptot= corminFAM([mship(eng(i),E1) mship(zcr(i),Z3)],J1,classrng); %(E1,Z3;J1)
fuzout = fuzout + temptot;
rulecnt(9) = rulecnt(9) + sum(temptot);

temptot= corminFAM([mship(eng(i),E2) mship(zcr(i),Z3)],J1,classrng); %(E2,Z3;J1)
fuzout = fuzout + temptot;
rulecnt(10) = rulecnt(10) + sum(temptot);

temptot= corminFAM([mship(eng(i),E3) mship(zcr(i),Z3)],J2,classrng); %(E3,Z3;J2)
fuzout = fuzout + temptot;
rulecnt(11) = rulecnt(11) + sum(temptot);

temptot= corminFAM([mship(eng(i),E4) mship(zcr(i),Z3)],J2,classrng); %(E4,Z3;J2)
fuzout = fuzout + temptot;
rulecnt(12) = rulecnt(12) + sum(temptot);
```

```
mptot= corminFAM([mship(eng(i),E1) mship(zcr(i),Z4)],UV,classrng); %(E1,Z4;UV)
zout = fuzout + temptot;
lecnt(13) = rulecnt(13) + sum(temptot);

mptot= corminFAM([mship(eng(i),E2) mship(zcr(i),Z4)],J1,classrng); %(E2,Z4;J1)
zout = fuzout + temptot;
lecnt(14) = rulecnt(14) + sum(temptot);

mptot= corminFAM([mship(eng(i),E3) mship(zcr(i),Z4)],J1,classrng); %(E3,Z4;J1)
zout = fuzout + temptot;
lecnt(15) = rulecnt(15) + sum(temptot);

mptot= corminFAM([mship(eng(i),E4) mship(zcr(i),Z4)],J1,classrng); %(E4,Z4;J1)
zout = fuzout + temptot;
lecnt(16) = rulecnt(16) + sum(temptot);

%% fuzzy centroid defuzzification %%%
lass(i) = sum(classrng.*fuzout)/sum(fuzout);          % (Kosko 8-19)
isp(['i = ' num2str(i) '//' num2str(2*numframes)]);
isp(['class(i) = ' num2str(class(i))]);

id

C = [LPC pp];
lass = round(class);
lass = reshape(class,2,numframes)';
pitch = find(pp == 0);
lass(nopitch,:) = zeros(length(nopitch),2);

ave fuzME

 The Transmit side is everything down to this point. %%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 From here down is the Receiver side %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

hatN = rcanal4L(LPC,N,class,fs,filename,runfile);
fileN = ['fuzME' filename num2str(zonk)];
isp('storing output file')
utsound(shatN,pfileN);

ave fuzME
```

```
%used for mixed excitation 4-level implementation
%excitations made at more than once per frame.

function    shatN = rcanal4L(LPC,N,class,fs,filename,runfile)
% LPC matrix arranged [Gain | LPC-coefficients | Pitch-period]
% N is frame length (number of samples per frame)

[numframes,c] = size(LPC);
[numframes,subfrms] = size(class);
disp('synthesizing speech')
numer = 1;

Z = zeros(c-3,1);
fvp = max(LPC(1:10,1));mix = 0;
lastpulse = 0;

for frame = 1:numframes
        %Determine estimate of Full voiced power
        if (frame == 1)           % 1st frame
           fvp = max(LPC(1:10,1));
           ifvp = 1;
        else
           fvp = max([LPC(frame,1) fvp*exp(-1.3*N/fs*(frame-ifvp)) ]);
           if (fvp == LPC(frame,1))
               ifvp = frame;
           end
        end
 fvpv(frame) = fvp;
        dbdiff = 10*log10(fvp)-10*log10(LPC(frame,1));
        %Set noise mix ratio
        if (dbdiff < 6)
            mix = 0.8;
        elseif (dbdiff > 18)
            mix = 0.5;
        else
            mix = -0.025*dbdiff + 0.95;
        end
mixv(frame) = mix;

    %Generate pulse sequence
    clear pindex;
  if (LPC(frame,c) == 0)
    rand('normal')
    ex = 0.33*rand(N,1);
  else
    if frame == 1
       pindex = [1:LPC(frame,c):N]';
    else
       pindex = [LPC(frame-1,c)-lastpulse:LPC(frame,c):N]';
    end
    sfrind = fix(pindex/(N/subfrms)) + 1;
    sfrind2 = rem(pindex,N/subfrms);

    wnind = find(class(frame,:)==0);        %subframes classed as UV
    j1ind = find(class(frame,:)==1);        %subframes classed as J1
    j2ind = find(class(frame,:)==2);        %subframes classed as J2
```

```
  voind = find(class(frame,:)==3);        %subframes classed as VO

  j1loc = []; j2loc = []; voind = [];

  for i = 1:length(j1ind)
       j1loc = [j1loc; find(sfrind==j1ind(i))];
  end

  for i = 1:length(j2ind)
       j2loc = [j2loc; find(sfrind==j2ind(i))];
  end

  for i = 1:length(voind)
       voloc = [voloc; find(sfrind==voind(i))];
  end

  rand('uniform')
  jitter1 = LPC(frame,c)*0.05*(1-2*rand(length(j1loc),1));
  jitter2 = LPC(frame,c)*0.03*(1-2*rand(length(j2loc),1));
  jitter3 = LPC(frame,c)*0.01*(1-2*rand(length(voloc),1));

  %pulse locations in J1 subframes
  sfrind2(j1loc) = round(sfrind2(j1loc) + jitter1);

  %pulse locations in J2 subframes
  sfrind2(j2loc) = round(sfrind2(j2loc) + jitter2);

  %pulse locations in VO subframes
  sfrind2(voloc) = round(sfrind2(voloc) + jitter3);

  index = find(sfrind2<1 & sfrind~=1);
  sfrind(index) = sfrind(index) -1;
  sfrind2(index) = N/subfrms - sfrind2(index);
  if(sfrind2(1)<1) sfrind2(1) = 1; end

%  pindexes = [pindex sfrind sfrind2]
  ex = zeros(N/subfrms,subfrms);

  for subframe = 1:subfrms
   if isempty(find(sfrind==subframe))
   else
   ex(sfrind2(find(sfrind==subframe)),subframe) =
                                        ones(length(find(sfrind==subframe)),1);
   end
      for i = 1:length(wnind)
           if wnind(i) == subframe
               rand('normal')
               ex(:,subframe) = 0.33*rand(N/subfrms,1);
           end
      end
  end
  ex = ex(:); ex = ex(1:N);
end
  index = find(ex~=0);
  lastpulse = N - index(length(index));
```

```
        %Make excitation sequence
        %G = -7/3*mix + 2.0667;
        G = -mix + 0.8;
        b = 1; a = b*G^2;
        rand('normal');w = rand(N,1);
        excit = G*filter([1 -b],1,w) + filter([1 a],1,ex);

        [shatN(:,frame),Z] = filter(numer,LPC(frame,2:c-1),LPC(frame,1)*excit, Z);

end

shatN = shatN/max(max(abs(shatN)));
shatN = shatN(:);


return
```

```
nction   window = rwind(data,N)


isp('Windowing into frames')
umrows, numcols] = size(data);
mframes = floor(numrows/N);
indow = reshape(data(1:numframes*N),N,numframes);

 for hamming window
 ham = hamming(N);
 for i = 1:numframes
       window(:,i) = ham.*window(:,i);
 end
eturn
```

```
function      R = stcorr(datamat,p)

disp('correlating')
[r,c] = size(datamat);
R=[];
for j = 0:p;
    if (j+1 ~= r)
      R(j+1,:) = sum(datamat(1:r-j,:).*datamat(j+1:r,:) );
    else
      R(j+1,:) = datamat(1:r-j,:).*datamat(j+1:r,:);
    end
end
return
```

```
function [sigma,a,gamma]=levinson(R,order)
%  LEVINSON(R,p) calculates the p-order AR prediction filter parameters
%                from the given ensemble of M correlation coefficient
%                vectors arranged along the columns of R:
%
%                        R(0,1)    R(0,2)  ...  R(0,M)
%                        R(1,1)    R(1,2)  ...  R(1,M)
%                          :         :            :
%                        R(p+1,1)  R(p+1,2)...  R(p+1,M)
%
%                If R is a vector of correlation coefficients, orientation
%                can be in either direction.
%
%  Returns: [sigma a gamma]
%
%           NOTE:   For the Levinson recursion, the correlation vectors are
%                   obtained from the first row of a TOEPLITZ correlation
%                   matrix.

%  By: Chris G. Kmiecik, LT USCG, 25 May, 1990.

[r Cols]=size(R);                                   %
if (r == 1)                                         % Re-orient if only one
  R=R.';                                            % vector has been given
  Cols=1;                                           % in a row rather than
end;                                                % a column.

if nargin == 1
  order=length(R(:,1))-1;
end

a=ones(1,Cols); b=a; sigma=a.*R(1,:);               % Initialize recursion.

for p=1:order
   r=R(2:p+1,:);                                    %    Compute Delta
   ar=a(p:-1:1,:);                                  % For the first iteration,
                                                    % 'r' and 'ar' will only
   if (p ~= 1)                                      % contain a single row;
     delta=sum(r.*ar);                              % SUM would sum across the
   else                                             % row so for the first
     delta=r.*ar;                                   % iteration, the sum is omitted
   end;                                             % to maintain the columns of 'a'.

   gamma(p,:)=delta./sigma;                         % A matrix of gamma values will
   sigma=sigma-gamma(p,:).*conj(delta);             % be returned.

   for k=2:p                                        % The first row of 'a' remains
     a(k,:)=a(k,:)-gamma(p,:).*ar(k-1,:);           % the same.  The last row of
   end;                                             % 'a' becomes minus gamma.
   a(p+1,:)=-gamma(p,:);                            %
end;

return;
```

```
unction    res = txanal(LPC,window)

,c] = size(LPC);
enom = 1;
 = zeros(c-2,1);

or i = 1:r
   [res(:,i),Z] = filter(LPC(i,2:c).',denom,window(:,i), Z );
  end

eturn
```

```
unction Ldata = Lpass(data);

b,a] = butter(8,600/4000);
r,c] = size(data);
or i = 1:c
   Ldata(:,i) = filter(b,a,data(:,i));
nd
eturn
```

```
unction  CLres = clip(res,N)
CLres  center clips the given residual to 68% of the peak value.

isp('clipping')
hird = floor(N/3);
ax13 = max(abs(res(1:third,:)));
ax33 = max(abs(res(N-third:N,:)));
%set clipping level
l = [(max13<=max33)].*(0.68*max13) + [(max33<=max13)].*(0.68*max33);
L = ones(N,1)*cl;
Lres = [(res>CL)].*(res-CL) + [(res< -CL)].*(res+CL); %clip the signal

eturn;
```

```
function pp = pitchper(data1,data2)
%data1 is short time autocorrelations of residual
%data2 is short time autocorrelation of original speech

% The original speech is used in cases where the residual does not
% provide good results.

[N,numframes] = size(data1);

pp = zeros(numframes,1);

for i = 1:numframes          %find pitch period for each frame
   index = find(data1(N/20:N,i)>0.35)+N/20-1;
   if (isempty(index))
       pp(i) = 0;
   else
       pp(i) = min(find(data1(:,i)==max(data1(index,i))));
       if ( data1(pp(i),i) < 0.4)
           index = find(data2(N/20:N,i)>0.35)+N/20-1;
           if (isempty(index))
              pp(i) = 0;
           else
              pp(i) = min(find(data2(:,i)==max(data2(index,i))));
           end
       end

       if (pp(i) < 25)
           pp(i) = 0;
       end
   end

end
pp = medfilt(pp,3);
pp = round(pp);
return;
```

```
unction  output =  medfilt(input,filtsz)

edfilt(input,filtsz) :  applies a median filter of size filtsz to the data vector
                        input.

r,c] = size(input);
ad = ones(1,filtsz-1)*input(1);

f (r==1)
  input = [pad input];
  output = zeros(1,c);
lseif (c==1)
  input = [pad' ; input];
  output = zeros(r,1);
lse
  error('input must be a vector')
nd

or i = filtsz:length(input)
   output(i-filtsz+1) = median(input(i-filtsz+1:i));
nd

eturn
```

```
function    eng = enrgy(window)

[N,numframes] = size(window);

eng = zeros(2*numframes,1);

%Overlap Frames
frame = 1;
for i = 1:2*numframes
     if i == 1
        wind(:,i) = window(:,i);
     elseif i == 2*numframes
        wind(:,i) = window(:,numframes);
     elseif rem(i,2) == 0
        wind(:,i) = [window(N/4+1:N,frame);window(1:N/4,frame+1)];
        frame = frame + 1;
     else
        wind(:,i) = [window(0.75*N+1:N,frame-1);window(1:0.75*N,frame)];
     end
end

eng = sum(wind.^2);

return
```

```
nction   zcr = zerocrs(window)

,numframes] = size(window);
r = zeros(2*numframes,1);

verlap Frames
ame = 1;
r i = 1:2*numframes
    if i == 1
       wind(:,i) = window(:,i);
    elseif i == 2*numframes
       wind(:,i) = window(:,numframes);
    elseif rem(i,2) == 0
       wind(:,i) = [window(N/4+1:N,frame);window(1:N/4,frame+1)];
       frame = frame + 1;
    else
       wind(:,i) = [window(0.75*N+1:N,frame-1);window(1:0.75*N,frame)];
    end
d

ind zero crossings
r i = 1:2*numframes
  zcr(i) = sum(floor((abs(diff(sign(wind(:,i))))./2)));
d

turn
```

```
function    SET = mkset(Llo,Lhi,Rhi,Rlo)

%SET = mkset(Llo,Lhi,Rhi,Rlo)
%defines fuzzy set parameters when given trapezoidal breakpoints:
%
%                              xxxxxxxxxxxxxx
%                           x                 x
%                        x                       x
%                     x                             x
%                   Llo    Lhi             Rhi    Rlo
%
%      SET  =  | Llo   Lm |
%              | Lhi   Lb |
%              | Rhi   Rm |
%              | Rlo   Rb |


%slopes
if Lhi == Llo
  Lm = 0;
else
  Lm = 1/(Lhi-Llo);
end
if Rhi == Rlo
  Rm = 0;
else
  Rm = 1/(Rhi-Rlo);
end

%y-intercepts
Lb = 1-Lm*Lhi;
Rb = 1-Rm*Rhi;

SET = [Llo Lm; Lhi Lb; Rhi Rm; Rlo Rb];
return;
```

```
nction   m = mship(x,SET)

m = mship(x,SET)
determines the membershiphood of the value x in the fuzzy set SET.
SET is defined as in mkset.m

 x >= SET(2,1) & x <= SET(3,1)
 m = 1;
seif x >= SET(1,1) & x <= SET(2,1)
 m = SET(1,2)*x + SET(2,2);
seif x >= SET(3,1) & x <= SET(4,1)
 m = SET(3,2)*x + SET(4,2);
se
 m = 0;
d

turn
```

```
unction   fuzout = corminFAM(antecedant,consequent,outrange)


r,c] = size(consequent);

uzout = zeros(outrange);
or i = 1:length(outrange)
  for j = 1:2:c

     %temp(j) = min(max(antecedant),mship(outrange(i),consequent(:,j:j+1)));
     % outer min is for corr-min operation;
     %inner max assumes OR combination of antecedants.

     temp(j) = min(min(antecedant),mship(outrange(i),consequent(:,j:j+1)));
     %outer min is for corr-min operation;
     %inner min assumes AND combination of antecedants.


  end
  fuzout(i) = min(temp); %this min assumes AND combination of consequents.
nd

eturn;
```

```
function SETS = findsets(parameter,wts,numsets)

rangehi = max(parameter);
rangelo = min(parameter);
setsize = (rangehi - rangelo)/numsets;


%initial set boundries
setbounds = zeros(1,numsets+1);
setbounds(1) = rangelo;
for i = 2:numsets+1
   setbounds(i) = setbounds(i-1) + setsize;
end

setcnts = zeros(1,numsets);
for i = 1:numsets
 setcnts(i) = length(find( parameter>=setbounds(i) & parameter<setbounds(i+1) ));
end
for k = 1:numsets-1
cntr = 0;
while ( abs(sum(setcnts)*wts(k) - setcnts(k)) > 1)

   cntr = cntr +1;
   if cntr>500 break;end

   delta = (setbounds(k+1) - setbounds(k))/(ceil(cntr/10)+1);
   %reassign set boundry
   if (sum(setcnts)*wts(k) < setcnts(k))
      setbounds(k+1) = setbounds(k+1) - delta;
   else
      setbounds(k+1) = setbounds(k+1) + delta;
   end

   %find new set counts
   setcnts = zeros(1,numsets);
   for i = 1:numsets
     setcnts(i) = length(find(parameter>=setbounds(i) & parameter<setbounds(i+1)));
     end
%disp(setcnts)
end
end

%figure breakpoints for sets
setpoints = zeros(numsets,4);
for i = 1:numsets-1
    setrng = setbounds(i+2) - setbounds(i);
    eps1 = setrng/8; eps2 = setrng/16;
    if (setbounds(i+1)-eps2 > setpoints(i,2))
       setpoints(i,3) = setbounds(i+1) - eps2;
    else
       setpoints(i,3) = setpoints(i,2);
    end
    setpoints(i,4) = setbounds(i+1) + eps1;
    setpoints(i+1,:) = [(setbounds(i+1)-eps1) (setbounds(i+1)+eps2) 0 0];
    if (setpoints(i+1,1)<0)
```

```
    setpoints(i+1,1) = 0;
  end
rl
setpoints(numsets,3) = rangehi+eps1;
setpoints(numsets,4) = setpoints(numsets,3);

ake the sets
TS = zeros(4*numsets,2);
r i = 1:numsets
TS(i*4-3:i*4,:) = mkset(setpoints(i,1),setpoints(i,2),setpoints(i,3),setpoints(i,4));
d

turn
```

APPENDIX B
RESULTS OF SIMULATIONS

This appendix contains the plotted results from simulations of several configurations of the vocoders developed in the thesis. The figures are arranged as follows:

- Figure B.1 to Figure B.2 are from the standard LPC vocoder.
- Figure B.3 through Figure B.7 are from the ME vocoder showing the effects of different orders of the predictor filter and various frame lenghts.
- Figure B.8 to Figure B.9 are from the MME vocoder at typical values.
- Figure B.10 through Figure B.11 show the results of the four level MME vocoder.
- Figure B.12 through Figure B.13 show the results from the four level fuzzy based MME vocoder.
- Figure B.14 through Figure B.15 is from the five level fuzzy based MME vocoder.

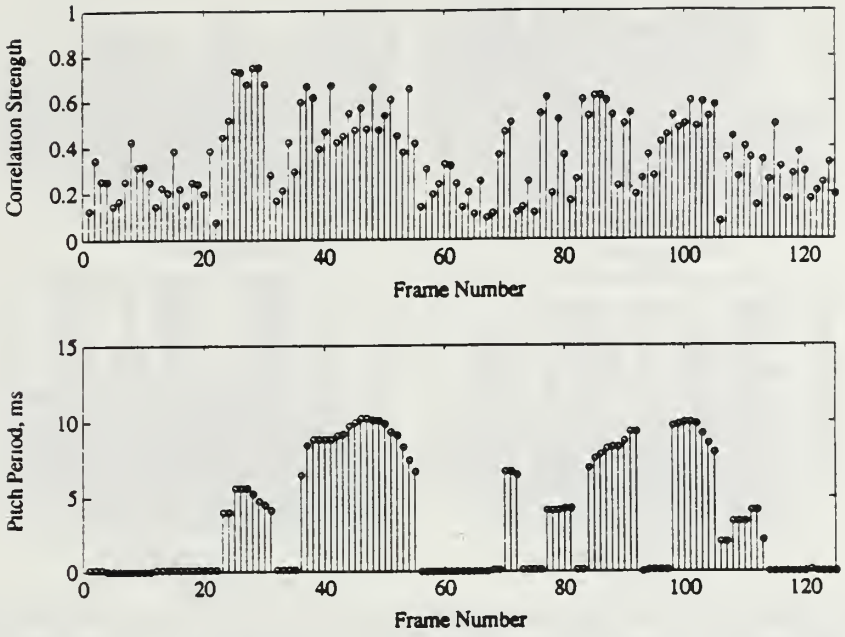

Figure B.1: Correlation strength and pitch period from the standard LPC vocoder. The speech sample is of a male speaker saying "Excuse me madame, would you care to dance?"

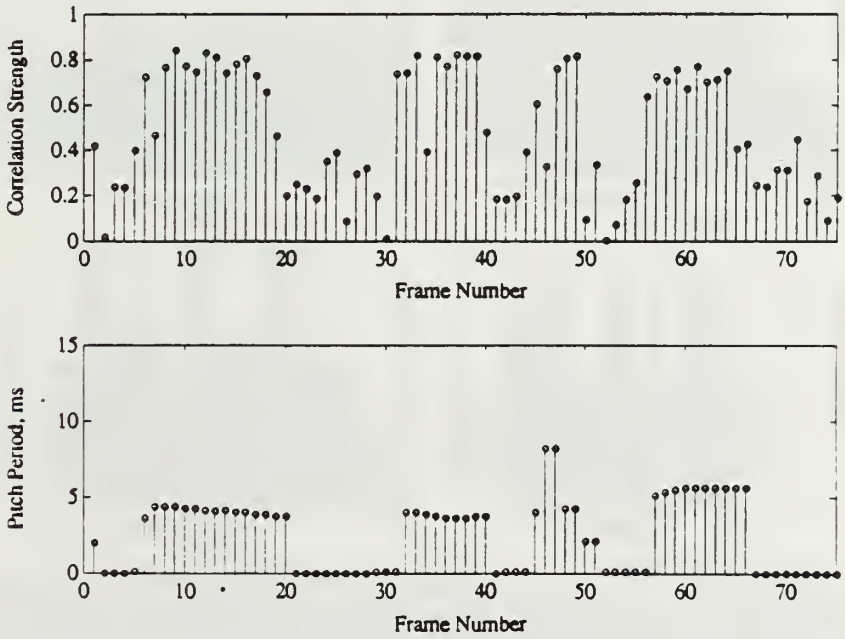

Figure B.2: Correlation strength and pitch period from the standard LPC vocoder. The speech sample is of a female speaker saying "No, I don't think so."

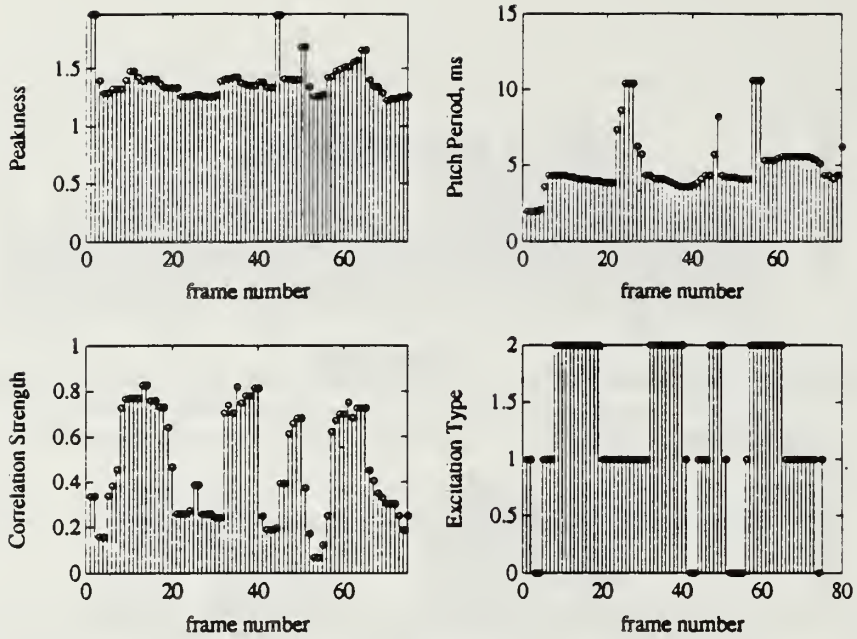

Figure B.3: Peakiness, correlation strength, pitch period, and excitation type from the Mixed Excitation vocoder, filter order = 8, frame length = 25 ms. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV for 1, and VO for 2.

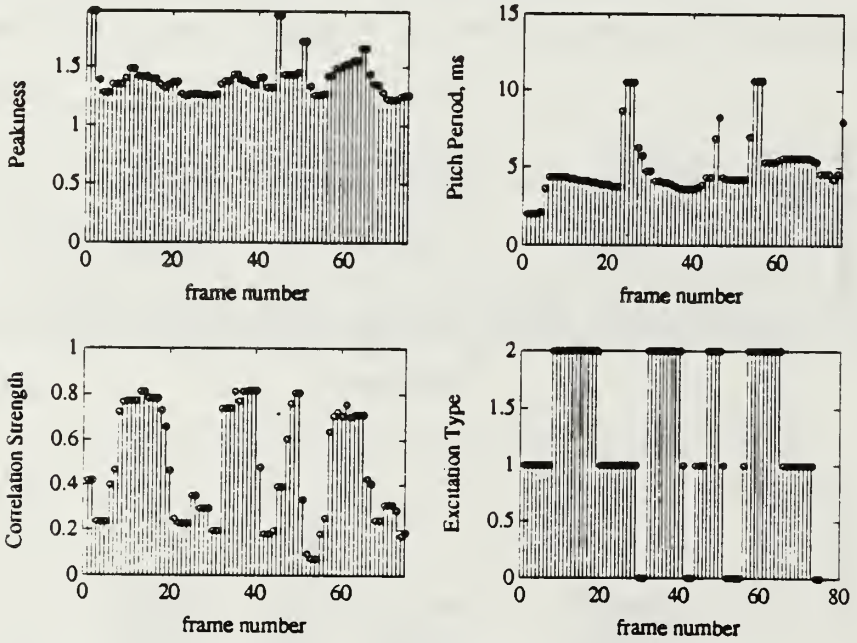

Figure B.4: Peakiness, correlation strength, pitch period, and excitation type from the Mixed Excitation vocoder, filter order = 10, frame length = 25 ms. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV for 1, and VO for 2.

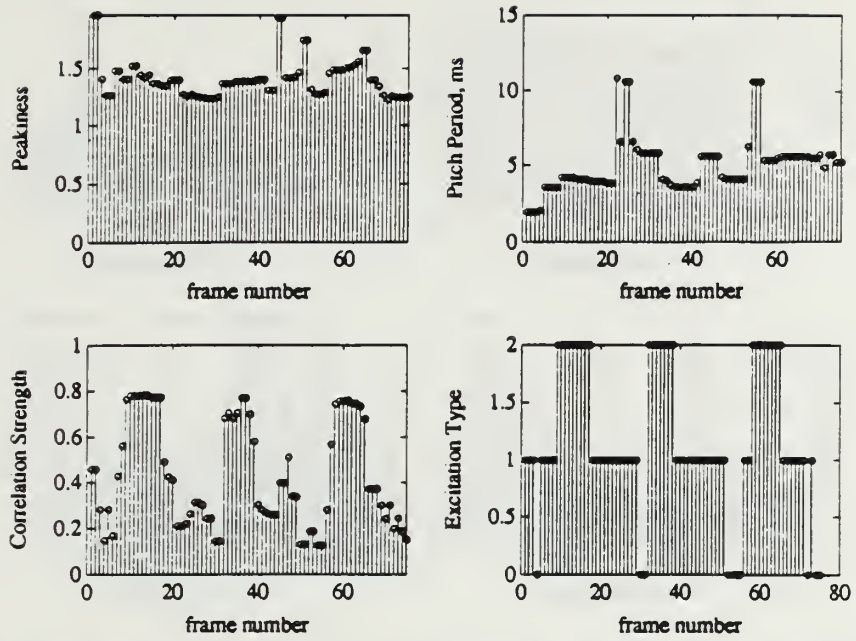

Figure B.5: Peakiness, correlation strength, pitch period, and excitation type from the Mixed Excitation vocoder, filter order = 16, frame length = 25 ms. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV for 1, and VO for 2.

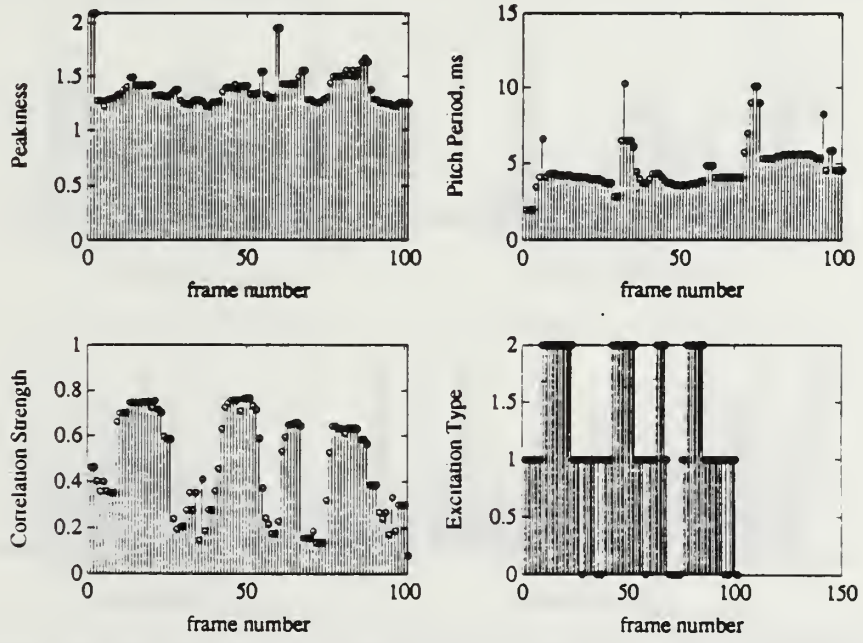

Figure B.6: Peakiness, correlation strength, pitch period, and excitation type from the Mixed Excitation vocoder, filter order = 10, frame length = 18.5 ms. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV for 1, and VO for 2.

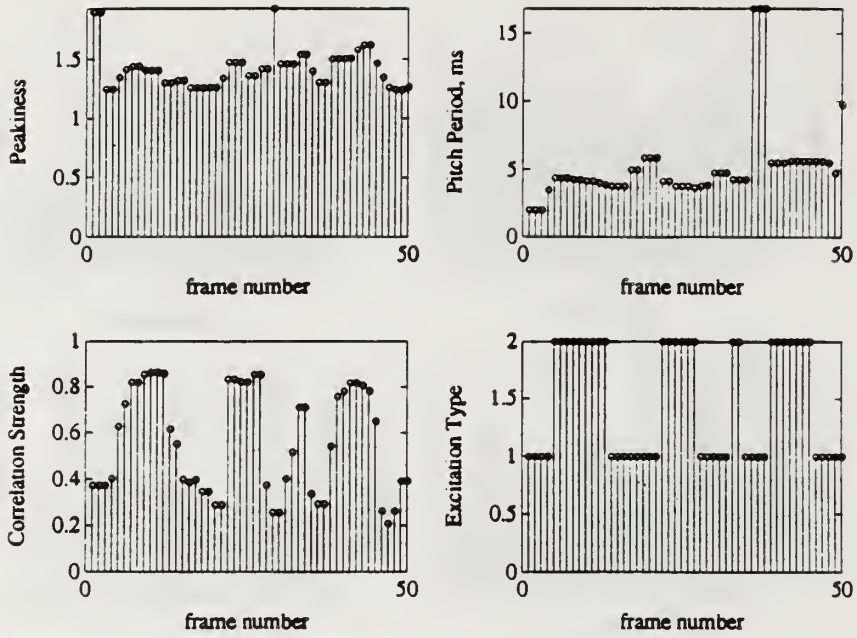

Figure B.7: Peakiness, correlation strength, pitch period, and excitation type from the Mixed Excitation vocoder, filter order = 10, frame length = 37.5 ms. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV for 1, and VO for 2.

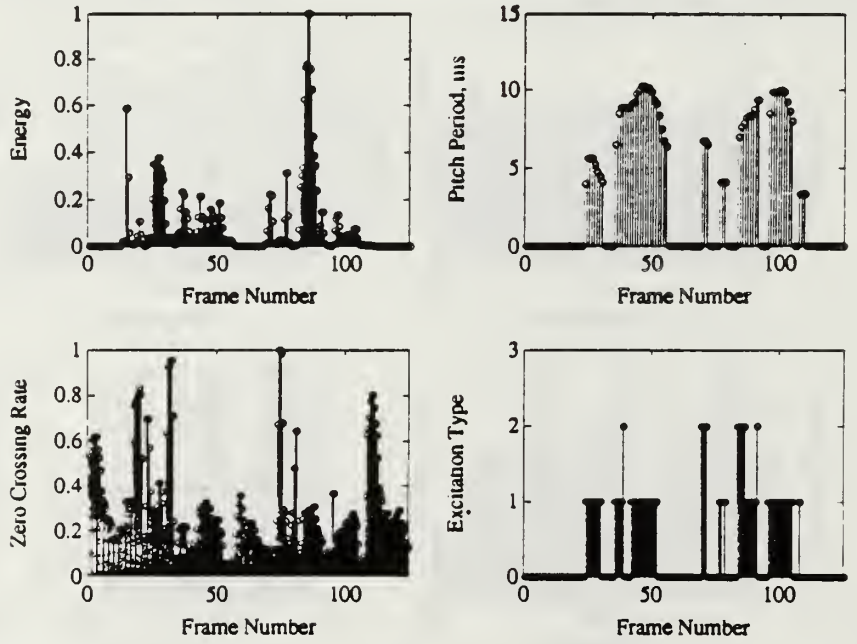

Figure B.8: Energy, zero crossing rate, pitch period, and excitation type from the MME vocoder. The speech sample is of a male speaker saying "Excuse me madame, would you care to dance?" The excitation types are UV for 0, JV for 1, and VO for 2.

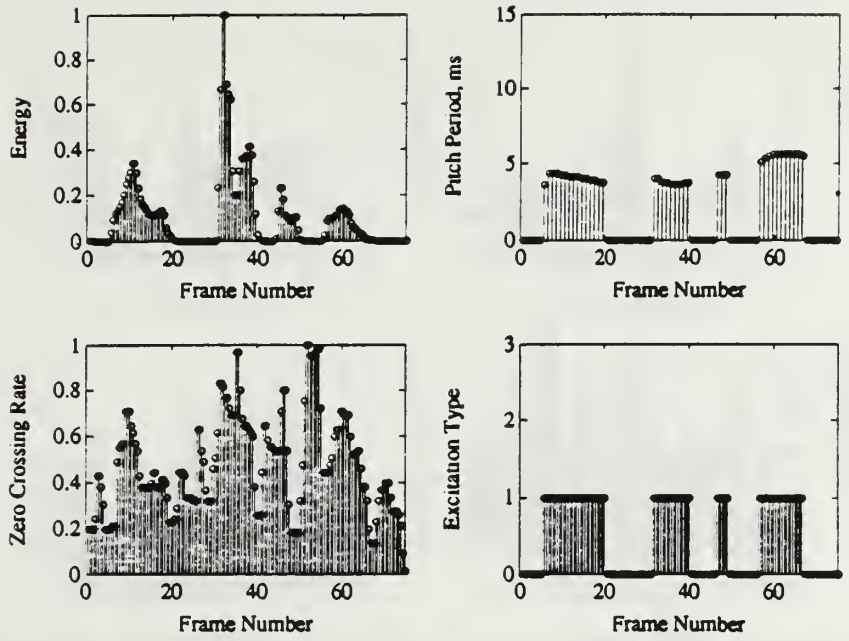

Figure B.9: Energy, zero crossing rate, pitch period, and excitation type from the MME vocoder. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV for 1, and VO for 2.

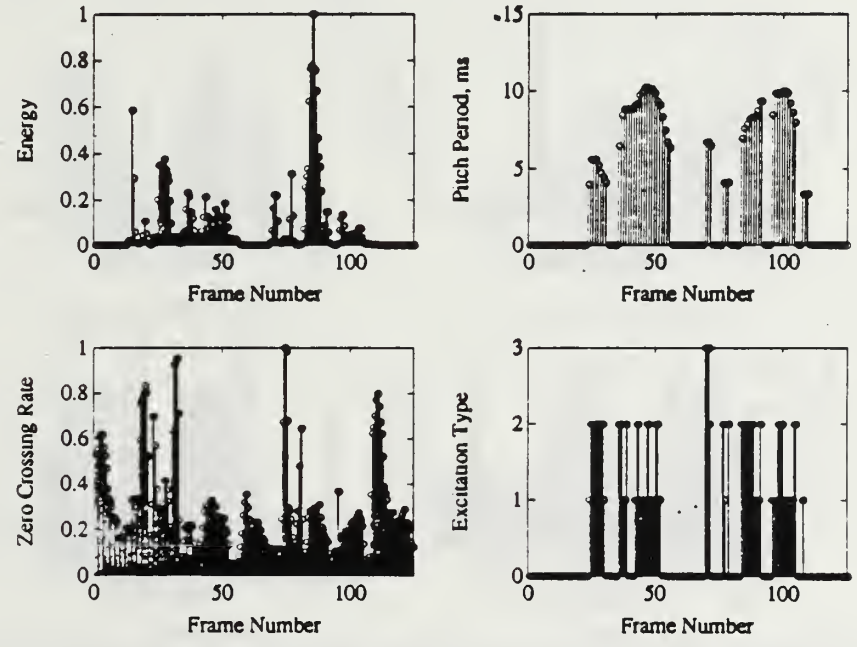

Figure B.10: Energy, zero crossing rate, pitch period, and excitation type from the four level MME vocoder. The speech sample is of a male speaker saying "Excuse me madame, would you care to dance?" The excitation types are UV for 0, JV1 for 1, JV2 for 2, and VO for 3.

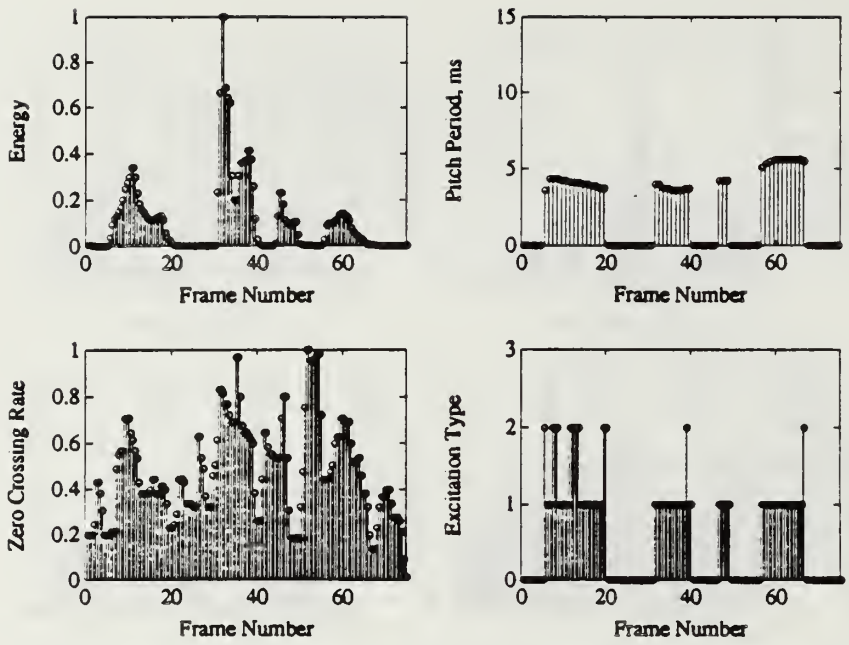

Figure B.11: Energy, zero crossing rate, pitch period, and excitation type from the four level MME vocoder. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV1 for 1, JV2 for 2, and VO for 3.

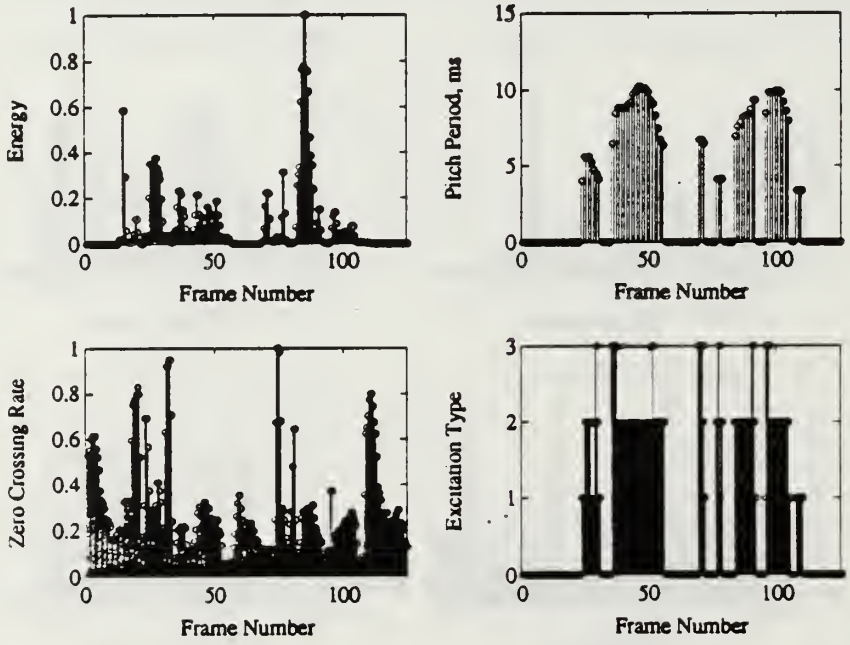

Figure B.12: Energy, zero crossing rate, pitch period, and excitation type from the four level fuzzy MME vocoder. The speech sample is of a male speaker saying "Excuse me madame, would you care to dance?" The excitation types are UV for 0, JV1 for 1, JV2 for 2, and VO for 3.

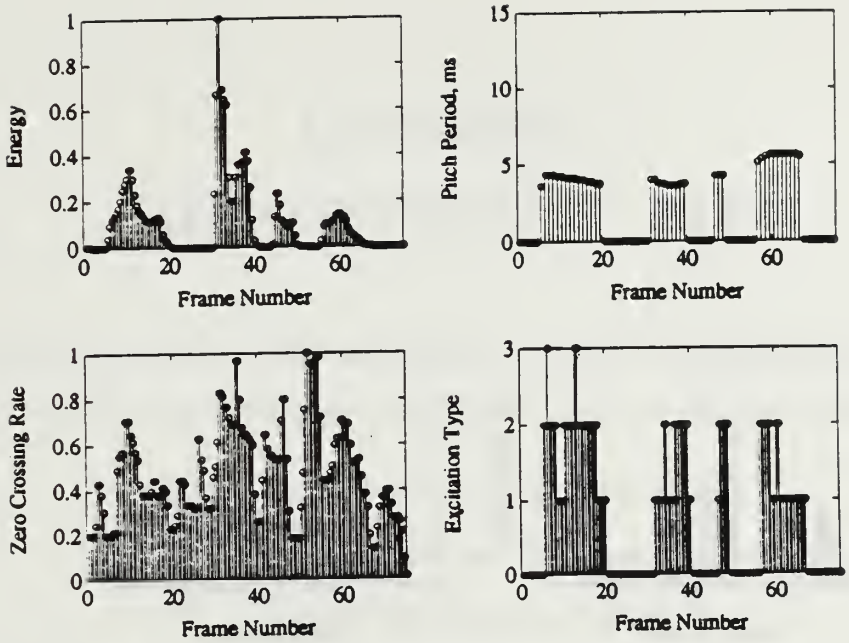

Figure B.13: Energy, zero crossing rate; pitch period, and excitation type from the four level fuzzy MME vocoder. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV1 for 1, JV2 for 2, and VO for 3.

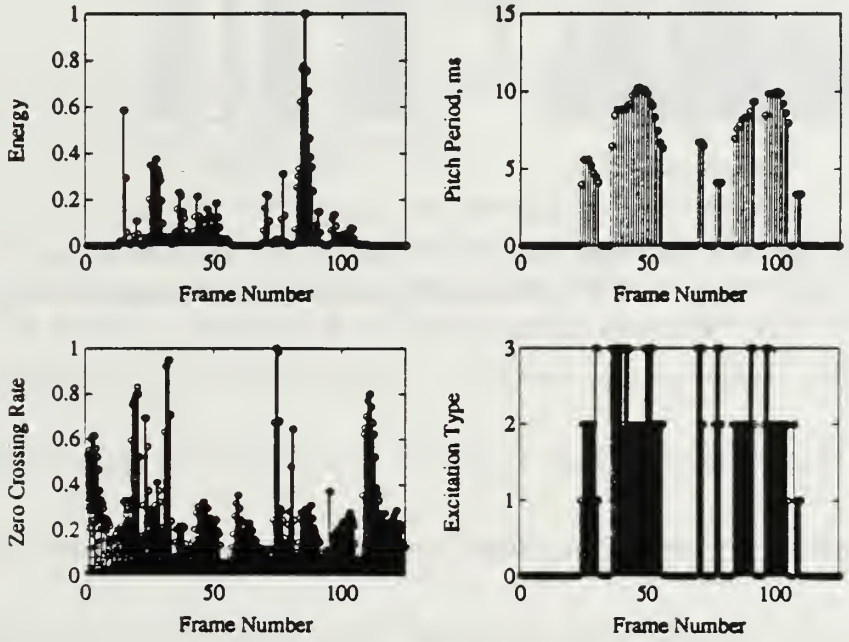

Figure B.14: Energy, zero crossing rate, pitch period, and excitation type from the five level fuzzy MME vocoder. The speech sample is of a male speaker saying "Excuse me madame, would you care to dance?" The excitation types are UV for 0, JV1 for 1, JV2 for 2, JV3 for 3, and VO for 4.

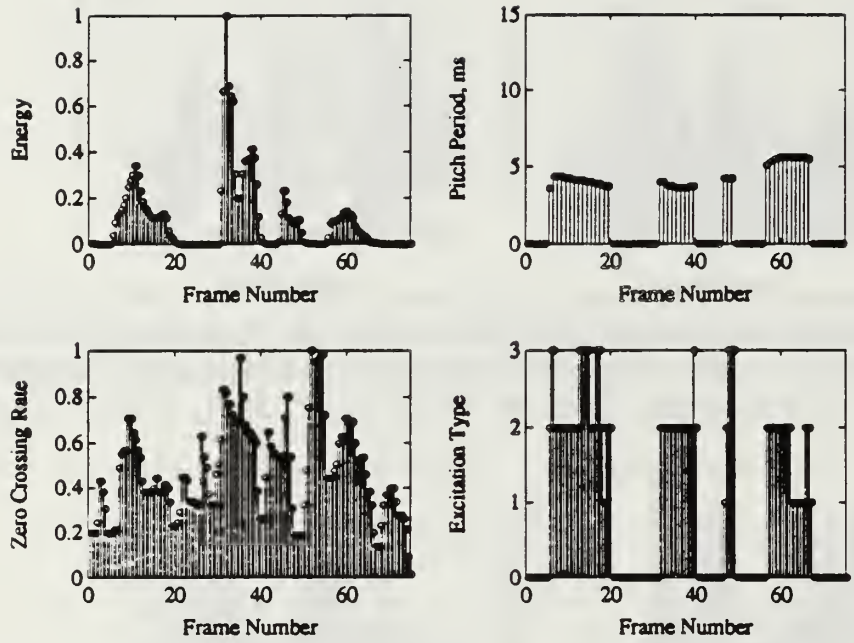

Figure B.15: Energy, zero crossing rate, pitch period, and excitation type from the five level fuzzy MME vocoder. The speech sample is of a female speaker saying "No, I don't think so." The excitation types are UV for 0, JV1 for 1, JV2 for 2, JV3 for 3, and VO for 4.

APPENDIX C
DEMONSTRATION TAPE

A demonstration cassette tape was made of the sythetic speech produced by several of the vocoders developed in the thesis. Copies of may be obtained from:

Professor Murali Tummala, Code EC/Tu
Department of Electrical and Computer Engineering
Naval Postgraduate School
Monterey, CA 93943

phone: (408)646-2645

Copies of the MATLAB files and audio files used in the development of this thesis are also available from the above address.

The speech samples on the tape are arrange as listed below.

1. Original speech, male speaker, "Asian cattle."
2. Standard LPC, 10 coefficients, 25 ms frames, male speaker, "Asian cattle."
3. ME vocoder, 8 coefficients, 25 ms frames, male speaker, "Asian cattle."
4. ME vocoder, 10 coefficients, 25 ms frames, male speaker, "Asian cattle."
5. ME vocoder, 16 coefficients, 25 ms frames, male speaker, "Asian cattle."
6. ME vocoder, 10 coefficients, 18.5 ms frames, male speaker, "Asian cattle."
7. ME vocoder, 10 coefficients, 37.5 ms frames, male speaker, "Asian cattle."

8. MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Asian cattle."

9. Four level MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Asian cattle."

10. Four level fuzzy MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Asian cattle."

11. Five level fuzzy MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Asian cattle."

12. Original speech, male speaker, "Baseball."

13. Standard LPC, 10 coefficients, 25 ms frames, male speaker, "Baseball."

14. ME vocoder, 10 coefficients, 25 ms frames, male speaker, "Baseball."

15. MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Baseball."

16. Four level MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Baseball."

17. Four level fuzzy MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Baseball."

18. Five level fuzzy MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Baseball."

19. Original speech, male speaker, "Excuse me madame, would you care to dance?"

20. Standard LPC, 10 coefficients, 25 ms frames, male speaker, "Excuse me madame, would you care to dance?"

21. ME vocoder, 10 coefficients, 25 ms frames, male speaker, "Excuse me madame, would you care to dance?"

22. MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Excuse me madame, would you care to dance?"

23. Four level MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Excuse me madame, would you care to dance?"

24. Four level fuzzy MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Excuse me madame, would you care to dance?"

25. Five level fuzzy MME vocoder, 10 coefficients, 25 ms frames, male speaker, "Excuse me madame, would you care to dance?"

26. Original speech, female speaker, "No, I don't think so."

27. Standard LPC, 10 coefficients, 25 ms frames, female speaker, "No, I don't think so."

28. ME vocoder, 8 coefficients, 25 ms frames, female speaker, "No, I don't think so."

29. ME vocoder, 10 coefficients, 25 ms frames, female speaker, "No, I don't think so."

30. ME vocoder, 16 coefficients, 25 ms frames, female speaker, "No, I don't think so."

31. ME vocoder, 10 coefficients, 18.5 ms frames, female speaker, "No, I don't think so."

32. ME vocoder, 10 coefficients, 37.5 ms frames, female speaker, "No, I don't think so."

33. MME vocoder, 10 coefficients, 25 ms frames, female speaker, "No, I don't think so."

34. Four level MME vocoder, 10 coefficients, 25 ms frames, female speaker, "No, I don't think so."

35. Four level fuzzy MME vocoder, 10 coefficients, 25 ms frames, female speaker, "No, I don't think so."

36. Five level fuzzy MME vocoder, 10 coefficients, 25 ms frames, female speaker, "No, I don't think so."

LIST OF REFERENCES

[1] L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

[2] Bart Kosko. *Neural Networks and Fuzzy Systems, a Dynamical Systems Approach To Machine Intelligence.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1992.

[3] Abdollah Homaifar and Ed McCormick. A New Approach for the Design and Implementation of Fuzzy Controllers. In *The Twenty-fourth Southeastern Symposium on System Theory; The Third Annual Symposium on Communications, Signal Processing, Expert Systems, and ASIC VSLI Design*, pages 313–317, Los Alamitos, CA, March 1992. IEEE, IEEE Computer Society Press.

[4] Charles W. Therrien. *Discrete Random Signals and Statistical Signal Processing.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1992.

[5] Douglas O'Shaughnessy. *Speech Communication, Human and Machine.* Addison-Wesley Publishing Company, Reading, MA, 1987.

[6] J. Makhoul. Linear Prediction: A Tutorial Review. *Proceedings of the IEEE*, 63(4):561–580, April 1975.

[7] B. Atal and J. Remde. A new model of LPC excitation for producing natural-sounding speech at low bit rates. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 614–617. IEEE, March 1982.

[8] W. B. Kleijin, D. J. Krasinski, and R. H. Ketchum. Fast Methods for the CELP Speech Coding Algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(8):1330–1342, August 1990.

[9] D. J. Rahikka, T. E. Tremain, V. C. Welch, and J. P. Campbell, jr. CELP Coding for Land Mobile Radio Applications. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 465–468. IEEE, March 1991.

[10] J. Chen. High Qualiy 16 kb/s Speech Coding with a One-way Delay Less Than 2 ms. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 453–456. IEEE, March 1990.

[11] D. W. Griffin and J. S. Lim. Multiband Excitation Vocoder. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(8):1223–1235, August 1988.

[12] P. C. Meuse. A 2400 bps Multi-Band Excitation Vocoder. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 9–12. IEEE, March 1990.

[13] A. V. McCree and T. P. Barnwell III. A New Mixed Excitation LPC Vocoder. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 593–596. IEEE, March 1991.

[14] Lawrence R. Rabiner and Ronald W. Schafer. *Digital Processing of Speech Signals.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1978.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304–6145	2
2.	Dudley Knox Library, Code 52 Naval Postgraduate School Monterey, CA 93943–5002	2
3.	Department Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	1
4.	Professor Murali Tummala, Code EC/Tu Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	2
5.	Professor Charles W. Therrien, Code EC/Th Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943	1
6.	Dr. R. Madan, Code 1114SE Office of Naval Research 800 North Quincy Street Arlington, VA 22217-5000	1
7.	Mr. John Hager, Code 70E1 Naval Undersea Warfare Engineering Station Keyport, WA 98345	1

8.	Mr. Samuel J. Frazier, Code SY84 Naval Air Warfare Center Patuxent River, MD 20670	1
9.	Commanding Officer Attn: LT J. T. Moore, USCG USCG Electronics Engineering Center P. O. Box 60 Wildwood, NJ 08260-0060	2